Markus Eisenbach, Ronny Stricker, Daniel Seichter, Alexander Vorndran, Tim Wengefeld and Horst-Michael Gross

Speeding up Deep Neural Networks on the Jetson TX1

URN: urn:nbn:de:gbv:ilm1-2018200017

Publikation entstand im Rahmen der Veranstaltung:
CAPRI 2017, Workshop on Computational Aspects of Pattern Recognition an Computer Vision with Neural Systems, 14-19 May 2017, Anchorage, Alaska, USA

in: Int. WS CAPRI at Joint Conf. on Neural Networks (IJCNN), Anchorage, USA, 2017

Speeding up Deep Neural Networks on the Jetson TX1

Markus Eisenbach, Ronny Stricker, Daniel Seichter, Alexander Vorndran, Tim Wengefeld, and Horst-Michael Gross*

Neuroinformatics and Cognitive Robotics Lab,
Ilmenau University of Technology,
98694 Ilmenau, Germany.
markus.eisenbach@tu-ilmenau.de

Abstract. In recent years, Deep Learning (DL) showed new top performances in almost all computer vision tasks that are important for automotive and robotic applications. In these applications both space and power are limited resources. Therefore, there is a need to apply DL approaches on a small and power efficient device, like the NVIDIA Jetson TX1 with a powerful GPU onboard. In this paper, we analyze the Jetson's suitability by benchmarking the run-time of DL operations in comparison to a high performance GPU. Exemplary, we port a top-performing DL-based person detector to this platform. We explain the steps necessary to significantly speed up this approach on the device.

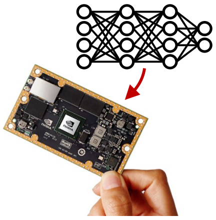
1 Introduction

In recent years, Deep Learning approaches have surpassed the performance of traditional computer vision methods by far for almost all image processing tasks that are essential for autonomous cars and mobile service robots. Examples are object recognition [27], scene understanding [8], person detection [6], and many more. All these DL approaches need extensive computational resources. Thus, usually they are processed on high performance GPUs. For neural network training, indeed, much computation power is indispensable to get the task done in a reasonable amount of time. Once trained, the deep networks can be applied on less powerful systems, but still need an adequate GPU to be fast.

Mobile robots have very tight power restrictions to guarantee an appropriate service accessibility time. Thus, high performance GPUs are typically unsuitable, since they are known for their high power consumption. Similar restrictions apply to the size and weight of components for autonomous cars, in order to increase the passenger compartment and carrying capacity as well as the car's fuel-efficiency. Bulky high performance PCs with powerful GPUs contradict this principle.

Luckily, in 2015 NVIDIA presented the Jetson TX1, an embedded system of the size of a credit card with a GPU onboard, that consumes less then ten

* This work has received funding from the German Federal Ministry of Education and Research as part of the SYMPARTNER project under grant agreement no. 16SV7218.



CPU	4-core ARM Cortex-A57 @ 1.9 GHz
GPU	256-core Maxwell @ 1 GHz
RAM	4 GB LPDDR4 (shared CPU + GPU)
Processing speed	1024 GFLOPS in float16 precision
Communication	Gigabit Ethernet, WiFi, Bluetooth, USB
Power consumption	< 10 W (peak 15 W in worst case)
Size	50×87 mm

Fig. 1. Basic data of NVIDIA Jetson TX1 platform used for Deep Learning.

watts (see Fig. 1). Therefore, this platform is perfectly suited for application on autonomous cars or mobile robots. For comparison, on our mobile robotic rehabilitation assistant [11] and our robotic companion for domestic use [13] we need two PCs with Intel core-i7 CPUs (i7-4770R, 4 Cores @ 3.2 GHz) to provide all the robot's services simultaneously, which together consume more than 170 watts.

In this paper, we show that computational expensive Deep Neural Network (DNN) computations can be outsourced to a Jetson TX1 with only very little communication overhead. Therefore, we show detailed run-time analyses of typical DL operations on this platform. Exemplary, we choose person detection as application, which is a central task for both autonomous cars (pedestrian recognition [4]) and mobile service robots (being aware of persons for keeping personal space [38], being polite in navigation [37], following a specific person [11], or identify the current user [7]).

In [6] we presented a DL-based person detector that surpassed the state of the art on the standard person detection benchmark dataset Caltech [4]. This neural net has relatively low memory requirements, which makes it a good candidate for porting it to the Jetson TX1. We will explain the steps necessary to speed up this approach on a Jetson TX1. Then we will show its application on a mobile robot, where it significantly outperforms traditional person detectors and other DL approaches.

2 Related Work

In recent years, low power consuming embedded devices with onboard GPU have gained increased attraction in a wide range of research fields. Deep neural networks heavily benefit from parallelization. Thus, they are perfectly suited for application on such devices.

2.1 Automotive Applications

The Jetson TX1 is widely spread among autonomous car research projects. It has been used for navigation tasks [32], sensor fusion and probabilistic tracking [19], semantic road scene segmentation based on dynamic programming [14], and

DL-based traffic sign recognition [25]. In [36] a DL-based semantic image segmentation is implemented on this embedded platform by applying the smaller SqueezeNet architecture instead of DL nets, that have many weights, to gain a speedup. The segmentation performance decreases due to the modifications but is still sufficient for self-driving cars. In [15] a related technique, called "distillation", is used for decreasing the size of a network in order to obtain computational savings.

2.2 Robotic Applications

Embedded platforms like Jetson TK1 and TX1 have been used on mobile robots for several tasks, such as the rectification of an omnidirectional image [33], particle-based monte carlo localization [29], SLAM [10], path planning [24], and speech processing [30]. In [16], sonar images of an underwater robot are classified by a DL approach on a TX1. They apply a retrained, but structurally unchanged YOLO detector [27] without further optimization.

In [21] a combination of a less powerful embedded platform on a mobile robot in combination with a cloud solution is proposed for Deep Learning. The onboard device processes just shallow neural networks. High accuracy can only be achieved by sending the computing job to a high performance server. In our opinion, this is not a feasible solution, since in many real-world environments WiFi might not be fast and reliable enough for unconstrained robot applications, as experienced e.g. in medical environments such as rehab clinics [11], in private apartments of elderly people [13], and in stores [12]. In these cases, the robot would not be able to provide service tasks that depend on DL-based modules. But a mobile service robot should be able to fulfill all of its tasks with high accuracy at any time. Therefore, DL operations should be performed onboard without the need for a cloud solution.

2.3 Person Detection

Also person detection has been done on a TK1 on a mobile robot. In [31] face detection is used to locate the user of a telepresence robot. In [2] persons are detected in 3D point clouds of a Kinect 1. These approaches, however, do not deploy Deep Learning.

In [1] classical computer vision based person detection approaches are ported to the Jetson TX1 achieving a huge speedup, up to 20 frames per second (HOG + LBP), due to the use of a GPU instead of a CPU. While the speed is very good, the accuracy is only mediocre.

Deep Learning approaches for person detection have also been ported to the Jetson TX1. In [40], an AlexNet [18] is retrained to detect skin color as indication for the presence of persons. This approach is not sufficient to detect persons robustly.

More often, general object detectors trained on ImageNet are applied. These include the class 'person', but are not fine-tuned on this specific task. In [20] a multi-scale wide residual inception network is applied for object detection on the TX1. It can process 16 rescaled frames of size 300×300 per second, but the detection results for the class 'person' are only mediocre. The YOLO object detector [27] is portet to the Jetson TK1 in [26]. It does not fit into the memory of the TK1. Therefore several strategies to reduce the network size are evaluated. Since fully connected (FC) layers need more than 80% of network's memory, the lack of memory could be handled by splitting the FC layers and processing parts of them sequentially. Other techniques, like decreasing the network size, degraded the performance. On the Jetson TK1, they processed 4 rescaled frames of size 448×448 per second. In [42], the YOLO detector was ported to a Jetson TX1 without modifications. They were able to process 12 frames of size 448×448 per second. In [22] the Faster R-CNN [28] approach based on VGG networks was ported to the Jetson TX1 without modifications. It is reported, that the detection for 1280×720 images "is near real-time frame rates".

All these approaches are relatively good general object detectors. However, for person detection they perform relatively poor (see Sec. 4.3).

2.4 Our Contribution

Summarized, none of these studies has analyzed the run-time of DL operations on the Jetson TX1 in detail. Only few of these studies have adapted their baseline DL approach in terms of computational savings for processing on the embedded platform. Most approaches take the neural nets as they are. In comparison, we analyze the run-time of DL operations and show, which additional optimizations speed up the computation without decreasing accuracy. As baseline, we use a specialized person detector from our previous work [6].

3 Speed up a Deep Learning based Person Detector on the Jetson TX1

3.1 Baseline Multi-Scale CNN Person Detector

For detecting persons at different scales, we build on our previous work [6] that set a new top mark on the most popular Caltech pedestrian detection benchmark. It uses a resolution pyramid in combination with three Convolutional Neural Networks (CNNs). Due to the use of multiple CNNs at different scales, the learned features are specific for the respective resolutions the particular CNNs are applied to, which improves the performance significantly. The network topologies used in the different stages are similar with the exemplary stage displayed in Fig. 2. The networks take raw pixels as input and predict whether the image patch shows a person or not. The networks were designed with a real-time application and an embedded device in mind. Thus, the largest net (Fig. 2) has relatively few weights (4M) to fit into the memory of a Jetson TX1, even when applied to a full-size image. More modern architectures known for their good performance on ImageNet would not fulfill this requirement. Additionally, the net is not overly deep and wide to avoid needless computations (see Fig. 2). In [6], the design choices are described in detail.

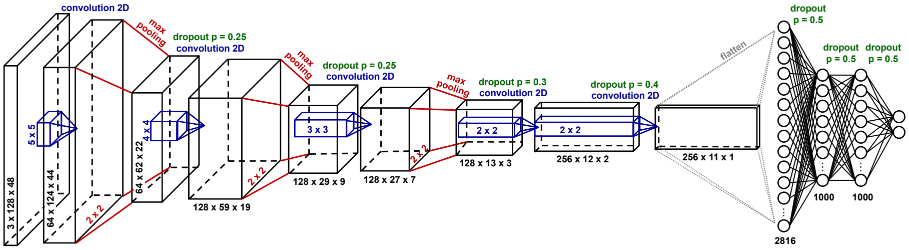


Fig. 2. CNN topology for detecting persons. The neural network consists of five convolutional layers with ReLU activation [23], three max-pooling layers, two fully connected layers with ReLU activation, and a softmax output layer. Dropout [34] is used for regularization.

The three CNNs were trained on a large and versatile dataset composed of 22 datasets from pedestrian detection and person re-identification domains. It contains cropped images showing persons (100,107 samples) and 628,636 non-person samples. The negative class includes random crops from non-person objects, typical false detections, and (on purpose) badly aligned crops showing only parts of persons. In this paper, we use the trained weights of [6]. Since the training dataset is versatile and generic, we do not need to retrain on a scenario specific dataset.

After the networks were trained, fully connected layers were converted to convolutional layers to be able to process images of any size without the need to shift a sliding window to several locations. Fig. 3 shows the processing chain of the application phase. Each of the CNNs calculates output maps for multiple scales of the resolution pyramid. When these classifications have been done, the

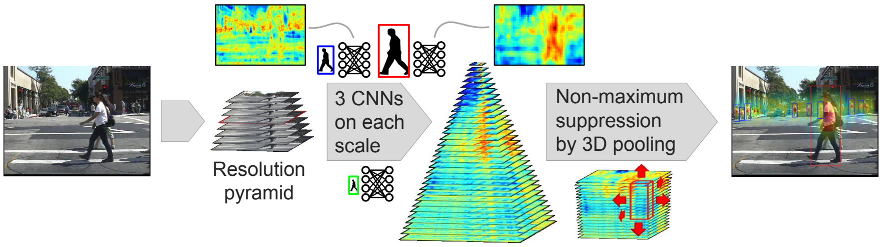


Fig. 3. Baseline approach [6]. Processing chain in the application phase. To create the resolution pyramid, the input image is scaled such that it is exactly halved in size after seven scales. Thus, each of the three CNNs has to process seven scales. The near scale CNN (red box) additionally processes smaller scales to detect larger persons. Exemplary network outputs are shown. High neural activations (shown in red) suggest that persons are present in that region of the image. The classification results are stacked and non-maximum suppression implemented as 3D pooling is applied to find the best fitting positions and scales for all persons in the scene.

full output pyramid can be constructed. Then, a 3D non-maximum suppression (NMS) is applied to find persons in the scene and at the best fitting scale. For NMS, we implemented an approximation of the mean-shift algorithm as 3D pooling. For algorithmic details we refer to [6].

For implementation, we used Keras [3] and Theano [35]. The network training was performed on a single NVIDIA GTX Titan X GPU in float32 precision.

3.2 Performance Analysis

The application phase on a NVIDIA GTX Titan X GPU took 0.231 seconds on average per image of size 640 × 480 if persons of a height of at least 80 pixels should be detected (reported in [6]). Persons, that are partly out of the image were not considered so far.

Hence, we extended this approach to also detect persons in front of the robot or pedestrians crossing the street in front of a car, where only the upper body is visible. This was achieved by zero padding the image below its bottom. The image size and, thus, also the computation time doubled. We also increased the maximum distance at which persons are detected. Therefore, the detector now searches for persons of height $75 - 927$ pixels on 26 scales.

When applied on the Jetson TX1, the run-time increased by a factor of 18 to 8.4 seconds. In this paper, we explain how to significantly speedup this DL-based detection approach by optimizing it for application on the NVIDIA Jetson TX1 without a loss in accuracy.

To figure out, what caused the significant slowdown, we first analyzed the run-time of DL operations on the Jetson TX1 in comparison to the high performance Titan X GPU (see Sec. 4.1). Convolutions, that account for 90% of the computations, are up to 11 times slower on the Jetson platform. Although, this explains most of the observed slowdown, the overall slowdown factor is still a lot higher. Therefore, we searched for additional slowdown factors that should be eliminated.

3.3 Optimizing the Detector for Processing on a Jetson TX1

Our analyzes have shown, that copy operations between CPU and GPU created a large overhead. Although, the Jetson's CPU and GPU share the same memory, making copy operations dispensable, the DL frameworks were not able to make use of this fact. To avoid unnecessary copy operations, we made sure, that everything is processed on the GPU, including non-DL operations such as construction of the image pyramid and postprocessing. Then, we optimized the computation graph in the Theano framework [35] by removing redundancy and by specifying exact shapes, which means, that all image and succeeding tensor sizes are set beforehand of processing. We can do this since the camera frame size will not change. These optimizations, that do not change any results, reduce the run-time by approximately 66%.

The run-time benchmarks of DL operations (see Fig. 4, Sec. 4.1) confirm, that float16 precision instead of float32 precision speeds up the computation significantly. We therefore checked, if using float16 precision instead of float32 precision changes the results significantly. The only part of the CNN that is extremely sensitive to a lower floating point precision is the gradient during backprobagation. During training float16 precision would significantly worsen the results and, thus, cannot be applied. In the application phase we observed, that the accuracy of the classifier does not change. None of the linear operations (Fig. 4 (a) – (c), (e) – (l)) are sensitive to precision. However, the outputs after the softmax operation loose floating point precision due to the exponential operation. That means, that non-maximum suppression (NMS) becomes hard, if not impossible, since positions near the optimal location of a detected person get equal scores. Therefore, operations after softmax need special treatment. We used the network's linear output before the softmax exponential operation for NMS. Using this trick, we observed only minor differences to the float32 version,

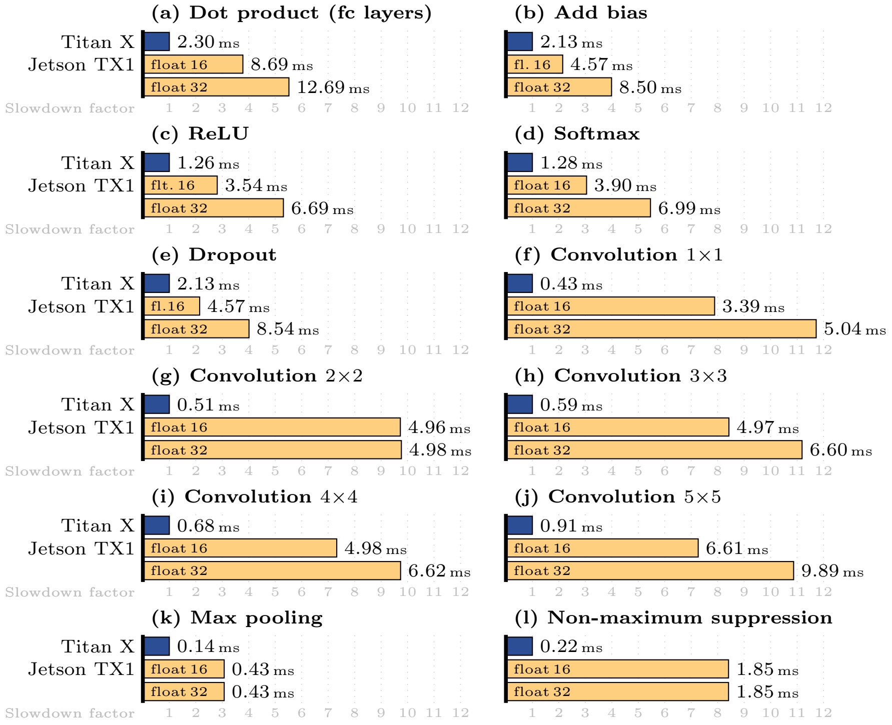


Fig. 4. Slowdown of typical Deep Learning operations on Jetson TX1 in comparison to GTX Titan X GPU. Absolute time measured in milliseconds is shown behind each bar. Times are averaged over 50 runs. (a) – (e) operations on 1000×1000 matrices, (f) – (j) valid convolutions on a 3-channel 320×240 image using 200 filters and a batch size of 1, (k) 2×2 max pooling on the same image size using stride 2×2, no padding and a batch size of 1, (l) 3×7×3 3D max pooling for non-maximum suppression on the same image size using stride 1×1×1, no padding and batch size 1.

while the accuracy did not change. Using float16 precision instead of float32 precision reduced the remaining run-time by approximately 25%.

To further speed up the computation, we introduced a ground plane constraint. By assuming that all persons must stand on the ground and by utilizing the extrinsic camera parameters, the image pyramid can be significantly reduced [17]. Since the relative position of the camera to the robot does not change over time, we can set all image sizes and succeeding tensor sizes beforehand. Again, when done properly, the accuracy does not decrease. Using this ground plane assumption, the remaining run-time was reduced by approximately 75%.

4 Experiments

4.1 Benchmarking Deep Learning on the Jetson TX1

To figure out, which DL operations mainly caused the significant slowdown on the Jetson TX1 in comparison to the high performance Titan X GPU, we first analyzed their run-time (Fig. 4).

Most critical are convolutions and rarely used and thus less optimized operations such as 3D pooling. Convolutions are significantly slower on the Jetson platform with a slowdown factor of up to 11. In our network, as in most other modern CNNs too, more than 90% of the computations are convolutions. This explains most of the observed slowdown. Fig. 4 also shows, that for large matrix and tensor sizes in common neural networks, all operations using float16 precision are faster than equivalent float32 precision operations.

4.2 Gained speedup

Tab. 1 shows the run-time for different stages of optimization. By optimizing the detector for processing on the Jetson TX1, we were able to speed up the run-time by a factor of 15 considering both preprocessing and DNN output computation (619.7 ms vs. 9,321.3 ms). At the same time, the accuracy did not decrease.

Table 1. Runtime of optimized detector on Jetson TX1

Detector	FloatX	Inp. scaling [ms]	Detection [ms]
Reference [6]	float32	930.1 ± 2.8	8,391.2 ± 15.3
Optimizations			
No overhead & float16	float16	46.1 ± 3.2	2,103.2 ± 5.9
+ Ground plane	float16	**32.1 ± 0.3**	**587.6 ± 1.8**

The steps described in Sec. 3.3 to reduce the run-time on an embedded platform are not specific for this detector, but can be applied to other Deep Learning approaches, too. Thus, we recommend to always check if the run-time can be reduced by:

- processing as much operations as possible on the GPU instead of the CPU to avoid overhead of copy operations,
- removing redundancies from the computation graph,
- specifying exact shapes to make all tensor sizes static,
- using float16 instead of float32 precision if accuracy does not drop,
- using assumptions to avoid needless computations.

4.3 Benchmarking the Person Detector on a Mobile Robot

We evaluated the person detector ported to this low power consuming small board on our mobile robot. Therefore, the Jetson TX1 was coupled to the main computer using its network interface, while data were exchanged using the robotics middleware MIRA [5]. In sum, data conversion, synchronization and communication for data transfer took only 3.36 ms per frame.

Tab. 2 shows the detection results on a recorded dataset [39].

Table 2. Detection performance of Computer Vision (CV) and Deep Learning (DL) approaches in a robotic application measured by miss rate (MR) for different false positives per image (FPPI). Additionally, the average number of frames that could be processed per second on a Jetson TX1 is reported.

Approach	**Type**	**MR @ 0.1 FPPI**	**MR @ 0.01 FPPI**	**Frame rate**
YOLO [27]	DL	0.386	0.918	**12** [42]
Faster R-CNN [28]	DL	0.321	0.734	2–3 [22]
Part-based HOG [9]	CV	0.273	0.603	1.5–2
Proposed [6]	DL	**0.115**	**0.287**	1.6

The proposed multi-scale detector based on [6] clearly outperforms the other approaches in accuracy. The other Deep Learning detectors (YOLO, Faster R-CNN) perform relatively poor in comparison. Their performance is in the proximity of the best classical approaches. This was also observed by Zhang *et al.* [41] on the Caltech dataset. In their extensive analysis, they found, that a Region Proposal Network ”specially tailored for pedestrian detection achieves competitive results as a stand-alone pedestrian detector. But surprisingly, the accuracy is degraded after feeding these proposals into the Fast R-CNN classifier.” One reason is, that small objects are not detected appropriately. We observed the same in our experiments. The second reason is the presence of unseen hard negative examples that trick most detectors to false detections. This issue can only be solved by training on a dataset including these difficulties, as we did in our previous work for the baseline CNN [6].

If a higher processing speed is desired, the detection can be divided and distributed onto multiple Jetson TX1. Four of these devices would still require less power than a single PC with a powerful CPU and would also require less space.

5 Conclusion

The small and low power consuming NVIDIA Jetson TX1 platform with a powerful GPU onboard makes it possible to apply top performing Deep Learning approaches on an autonomous car or a mobile robot. Thus, all DL solutions can be processed onboard. Exemplary, we showed how to port a DL-based person detector to this platform. Furthermore, we showed how to speed up the detector's run-time by factor 15. The result is a top performing DL-based person detector fast enough to replace the currently used CPU-based classical computer-vision-based detector on our robot. Our benchmark of typical DL operations will help other researches to estimate the run-time of Deep Learning approaches when applied on a Jetson TX1. Additionally, we have presented a list of generally applicable optimizations to speedup the computation on that device. In the qualitative evaluation on a robotic person detection benchmark dataset, the proposed detector clearly outperforms the state of the art including the popular DL-based object detectors YOLO and Faster R-CNN.

References

1. Campmany, V., Silva, S., Espinosa, A., Moure, J.C., Vázquez, D., López, A.M.: Gpu-based pedestrian detection for autonomous driving. In: Int. Conf. on Computational Science (ICCS). pp. 2377–2381. Elsevier (2016)
2. Carraro, M., Munaro, M., Menegatti, E.: A powerful and cost-efficient human perception system for camera networks and mobile robotics. In: Int. Conf. on Intelligent Autonomous Systems (IAS). pp. 485–497. Springer (2016)
3. Chollet, F.: Keras. `https://github.com/fchollet/keras` (2015)
4. Dollar, P., Wojek, C., Schiele, B., Perona, P.: Pedestrian detection: An evaluation of the state of the art. Pattern Analysis and Machine Intelligence, IEEE Transactions on 34(4), 743–761 (2012)
5. Einhorn, E., Langner, T., Stricker, R., Martin, Ch., Gross, H.M.: MIRA – middleware for robotic applications. In: Int. Conf. on Intelligent Robots and Systems (IROS). pp. 2591–2598. IEEE (2012)
6. Eisenbach, M., Seichter, D., Wengefeld, T., Gross, H.M.: Cooperative multi-scale convolutional neural networks for person detection. In: World Congress on Computational Intelligence (WCCI). pp. 267–276. IEEE (2016)
7. Eisenbach, M., Vorndran, A., Sorge, S., Gross, H.M.: User recognition for guiding and following people with a mobile robot in a clinical environment. In: Int. Conf. on Intelligent Robots and Systems (IROS). pp. 3600–3607. IEEE (2015)
8. Farabet, C., Couprie, C., Najman, L., LeCun, Y.: Learning hierarchical features for scene labeling. Transactions on Pattern Analysis and Machine Intelligence (TPAMI) 35(8), 1915–1929 (2013)
9. Felzenszwalb, P.F., Girshick, R.B., McAllester, D., Ramanan, D.: Object detection with discriminatively trained part-based models. Trans. on Pattern Analysis and Machine Intelligence (TPAMI) 32(9), 1627–1645 (2010)
10. Gong, Z., Li, J., Li, W.: A low cost indoor mapping robot based on tinyslam algorithm. In: Int. Geoscience and Remote Sensing Symposium (IGARSS). pp. 4549–4552. IEEE (2016)

11. Gross, H.M., Meyer, S., Scheidig, A., Eisenbach, M., Mueller, S., Trinh, T.Q., Wengefeld, T., Bley, A., Martin, C., Fricke, C.: Mobile robot companion for walking training of stroke patients in clinical post-stroke rehabilitation. In: Int. Conf. on Robotics and Automation (ICRA). IEEE (2017)
12. Gross, H.M., Boehme, H.J., Schroeter, Ch., Mueller, St., Koenig, A., Einhorn, E., Martin, Ch., Merten, M., Bley, A.: TOOMAS: Interactie shopping guide robots in everyday use – final implementation and experiences from long-term field trials. In: Int. Conf. on Intelligent Robots and Systems (IROS). pp. 2005–2012. IEEE (2009)
13. Gross, H.M., Mueller, St., Schroeter, Ch., Volkhardt, M., Scheidig, A., Debes, K., et al.: Robot companion for domestic health assistance: Implementation, test and case study under everyday conditions in private apartments. In: Int. Conf. on Intelligent Robots and Systems (IROS). pp. 5992–5999. IEEE (2015)
14. Hernandez-Juarez, D., Espinosa, A., Vázquez, D., López, A.M., Moure, J.C.: Gpu-accelerated real-time stixel computation. arXiv preprint arXiv:1610.04124 (2016)
15. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. In: Neural Information Processing Systems (NIPS) Workshop: Deep Learning and Representation Learning (2014)
16. Kim, J., Yu, S.C.: Convolutional neural network-based real-time rov detection using forward-looking sonar image. In: Autonomous Underwater Vehicles (AUV). pp. 396–400. IEEE/OES (2016)
17. Kolarow, A., Schenk, K., Eisenbach, M., Dose, M., Brauckmann, M., Debes, K., Gross, H.M.: APFel: The intelligent video analysis and surveillance system for assisting human operators. In: Int. Conf. on Advanced Video and Signal-Based Surveillance (AVSS). pp. 195–201. IEEE (2013)
18. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems (NIPS). pp. 1097–1105 (2012)
19. Laugier, C., Chartre, J., Perrollaz, M., Stein, S., et al.: Intelligent perception and situation awareness for automated vehicles. In: Conf. GTC Europe (2016)
20. Lee, Y., Kim, H., Park, E., Cui, X., Kim, H.: Wide-residual-inception networks for real-time object detection. arXiv preprint arXiv:1702.01243 (2017)
21. Leroux, S., Bohez, S., Verbelen, T., Vankeirsbilck, B., Simoens, P., Dhoedt, B.: Resource-constrained classification using a cascade of neural network layers. In: Neural Networks (IJCNN), 2015 International Joint Conference on. pp. 1–7. IEEE (2015)
22. Mhalla, A., Gazzah, S., Ben Amara, N.E., et al.: A faster r-cnn multi-object detector on a nvidia jetson tx1 embedded system: Demo. In: Int. Conf. on Distributed Smart Camera (ICDSC). pp. 208–209. ACM (2016)
23. Nair, V., Hinton, G.E.: Rectified linear units improve restricted boltzmann machines. In: Int. Conf. on Machine Learning (ICML). pp. 807–814 (2010)
24. Orozco-Rosas, U., Montiel, O., Sepúlveda, R.: Embedded implementation of a parallel path planning algorithm based on eapf for mobile robotics. In: Congreso Internacional en Ciencias Computacionales (CiComp). pp. 178–184 (2016)
25. Otterness, N., Yang, M., Rust, S., Park, E., Anderson, J.H., Smith, F.D., Berg, A., Wang, S.: An evaluation of the nvidia tx1 for supporting real-time computer-vision workloads. In: Real-Time and Embedded Technology and Applications Symposium (RTAS). IEEE (2017)
26. Rallapalli, S., Qiu, H., Bency, A.J., Karthikeyan, S., Govindan, R.: Are very deep neural networks feasible on mobile devices? Tech. rep., University of Southern California, TR 16-965 (2016)

27. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: Conf. on Computer Vision and Pattern Recognition (CVPR). pp. 779–788. IEEE (2016)
28. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: Towards real-time object detection with region proposal networks. In: Advances in Neural Information Processing Systems (NIPS). pp. 91–99 (2015)
29. Rud, M.N., Pantiykchin, A.R.: Development of gpu-accelerated localization system for autonomous mobile robot. In: Int. Conf. on Mechanical Engineering, Automation and Control Systems (MEACS). pp. 1–4. IEEE (2014)
30. Sansen, H., Torres, M.I., Chollet, G., Glackin, C., Petrovska-Delacretaz, D., Boudy, J., Badii, A., Schlögl, S.: The roberta ironside project: A dialog capable humanoid personal assistant in a wheelchair for dependent persons. In: Int. Conf. on Advanced Technologies for Signal and Image Processing (ATSIP). pp. 381–386. IEEE (2016)
31. Satria, M.T., Gurumani, S., Zheng, W., Tee, K.P., Koh, A., Yu, P., Rupnow, K., Chen, D.: Real-time system-level implementation of a telepresence robot using an embedded gpu platform. In: Design, Automation & Test in Europe Conf. & Exhibition (DATE). pp. 1445–1448. IEEE (2016)
32. Shimchik, I., Sagitov, A., Afanasyev, I., Matsuno, F., Magid, E.: Golf cart prototype development and navigation simulation using ros and gazebo. In: Int. Conf. on Measurement Instrumentation and Electronics (ICMIE). vol. 75. EDP Sciences (2016)
33. Silva, G., Reátegui, J., Rodriguez, P.: Fast omni-image unwarping on the jetson tk1. In: GPU Technical Conf. (GTC) (2015)
34. Srivastava, N., Hinton, G.E., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. Journal of Machine Learning Research 15(1), 1929–1958 (2014)
35. Theano Development Team: Theano: A Python framework for fast computation of mathematical expressions. arXiv preprint arXiv:1605.02688 (2016)
36. Treml, M., Arjona-Medina, J., Unterthiner, T., Durgesh, R., Friedmann, F., Schuberth, P., Mayr, A., Heusel, M., Hofmarcher, M., Widrich, M., Bodenhofer, U., Nessler, B., Hochreiter, S.: Speeding up semantic segmentation for autonomous driving. In: Neural Information Processing Systems (NIPS) (2016)
37. Trinh, Th.Q., Schroeter, Ch., Gross, H.M.: ”go ahead, please”: Recognition and resolution of conflict situations in narrow passages for polite mobile robot navigation. In: Int. Conf. on Social Robotics (ICSR). pp. 643–653. Springer (2015)
38. Weinrich, Ch., Volkhardt, M., Einhorn, E., Gross, H.M.: Prediction of human collision avoidance behavior by lifelong learning for socially compliant robot navigation. In: Int. Conf. on Robotics and Automation (ICRA). pp. 376–381. IEEE (2013)
39. Wengefeld, T., Eisenbach, M., Trinh, Th.Q., Gross, H.M.: May i be your personal coach? bringing together person tracking and visual re-identification on a mobile robot. In: Int. Symposium on Robotics (ISR). pp. 141–148. VDE (2016)
40. Yu, C.H., Lee, T., Chen, C.W., Hsieh, M.R., Fuh, C.S.: Human segmentation with nvidia-tx1. In: Conf. on Computer Vision, Graphics, and Image Processing (CVGIP). pp. 1–8. IPPR (2016)
41. Zhang, L., Lin, L., Liang, X., He, K.: Is faster r-cnn doing well for pedestrian detection? In: European Conf. on Computer Vision (ECCV). pp. 443–457. Springer (2016)
42. Zhang, W., Zhao, D., Xu, L., Li, Z., Gong, W., Zhou, J.: Distributed embedded deep learning based real-time video processing. In: Int. Conf. on Systems, Man, and Cybernetics (SMC). pp. 1945–1950. IEEE (2016)