TensorFlow Enabled Deep Learning Model Optimization for enhanced Realtime Person Detection using Raspberry Pi operating at the Edge*

Reenu Mohandas, Mangolika Bhattacharya, Mihai Penica, Karl Van Camp, and Martin J. Hayes

University of Limerick, Ireland
reenu.mohandas@ul.ie, mango.bhattacharya@ul.ie, mihai.penica@ul.ie, karl.vancamp@ul.ie, martin.j.hayes@ul.ie

Abstract. In this paper Quantization effects are assessed for a real time Edge based person detection use case that is based on the use of a Raspberry Pi. TensorFlow architectures are presented that enable the use of real-time person detection on the Raspberry Pi. The model quantization is performed, performance of quantized models is analyzed, and worst-case performance is established for a number of deep learning object detection models that are capable of being deployed on the Pi for real-time applications. The study shows that the inference time for a suitably optimized TensorFlow enabled solution architecture is significantly lower than for an unquantized model with only slight cost implications in terms of accuracy when benchmarked against a desktop implementation. An industrial standard floor limit value of greater than 70% is achieved on the quantized models considered with a reduced detection time of less than 3ms. The Deep Neural Network model is trained using the INRIA Person Detection benchmark Dataset.

Keywords: Person Detection · Edge Intelligence· Edge Computing · Model Optimization · Model Quantization.

1 Introduction

Person recognition subsystems have now reached a certain level of maturity in many autonomous detection systems. The detection subsystems range from computationally less expensive use cases like simple people counting that can still use Infra-Red (IR) systems or heat-map processing to identification problems with more complex surveillance applications. Such cognitive applications invariably depend on a deep learning framework for robust performance. The deep

* This work is supported by Accucode Europe Ltd. and Confirm, a Science Foundation of Ireland research center in Smart Manufacturing hosted by the University of Limerick.

learning frameworks will encompass aspects of representation learning, high-level abstraction of non-linear raw signal data and will contain an automatic feature extraction capability [24].

The demand for fast and robust person detection in indoor as well as outdoor use-cases is necessary in this accelerated urbanizing environment. Central to the use case is a requirement for the use of a TensorFlow enabled approach within a CNN (Convolutional Neural Network) framework. Neural networks that consist of chains of tensor operations, (geometric trans-formations, affine transformations, rotation, scaling and so on) are use cases that are attracting a lot of attention in the literature of late. Pre-processing of the input data and a suitable framework which can process the tensor data on a device like the Pi is necessary for the geometric interpretation of such operations in low cost commercial applications [8]. Hence TensorFlow is considered as a necessary inference step for person detection in the deep learning use case that is considered here. In this work 'TensorFlow lite' is considered for the inference processes on the Pi. The TensorFlow framework used here is an open-source framework developed for internal use by Google for machine learning but was later released under the Apache 2.0 open source license in the year 2015 [17].

This work also applies some recent advances in the study of the integration of deep learning frameworks to exploit the strong inductive biases that have been observed when applying neural networks to optimization or machine learning problems, [11] CNNs are an essential component in a deep learning framework used for detection and classification from image/video input. CNNs have been the preferred neural network-based approach to pixel-wise image segmentation over traditional image processing and computer vision techniques, especially in real-time person detection and identification problems [12]. Unlike the conventional image processing techniques which involve HOG, SVM or other gradient based methods, Deep Learning makes it easier to implement person detection due to its automatic feature extraction capabilities [5]. Transfer Learning have made Deep Learning more versatile as the base model trained on a sufficiently large dataset can be further used to find solutions of new problems with few steps of fine-tuning for the specific use-case.

The efficacy of such deep learning frameworks is always a focus of academic research. In this work use case performance is assessed across a GPU powered device and an Edge computing device with regard to latency calculations as opposed to the standard processing time requirements in an industry-specific application, and use of quantization approach for performance enhancement. This paper considers the efficiency vs latency of person detection on Edge Computing devices against a GPU accelerated device. This analysis is significant because of the advancement of Edge Intelligence, where every technology is swiftly moving into resource constrained devices, and there is an increased need to maintain robust performance.

The literature is focused briefly on the types of optimization that were previously used for model optimization. There is a comparison of how various deep learning object detection models respond to quantization.

This paper is organized as follows. In Section II, the concept of Edge Intelligence is discussed This work should be viewed very much as an example of AI on the Edge use case based on a Pi type infrastructure. Model optimization and compression efforts is considered in Section III. In Section IV, the experiment is explained, and the results are analyzed. Finally, we present some conclusions based on the results that have been obtained and some recommendations for future work.

2 Edge Intelligence

Conventional deep learning frameworks are bulky, consume hundreds of megabytes necessary for trained weight storage and the inclusion of a necessary inference process. For those deep learning models that rely on dense layers, the number of parameters can number in the billions [7]. This explosion in parameters makes it challenging for reduced instruction set embedded or mobile systems to perform such cumbersome calculations in real-time. This has motivated the use of so-called performance 'optimized' neural networks that are denoted as edge applications. The process of integrating edge computing and AI, termed as edge intelligence, has attracted significant attention in the literature of late [10].

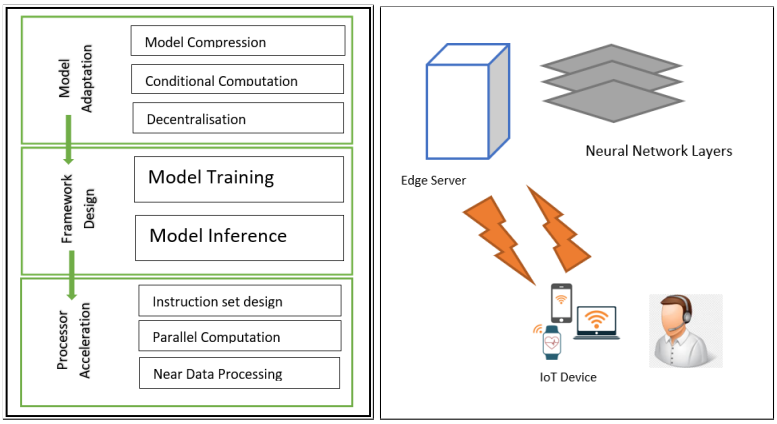

Fig. 1. A road map of edge intelligence on a top-down decomposition [10](left). Edge Computing structure for IoT Devices(right)

In [19], the peripheral control devices of industrial electronic systems often set up in the local ethernet is referred to as the edge computing infrastructure. Edge computing is a decentralized intelligent system with independent entities as per Kristiani et.al [18]. Edge computing has often been combined with IoT(Internet of Things)-based decentralized and distributed data capture infrastructure for advanced applications.

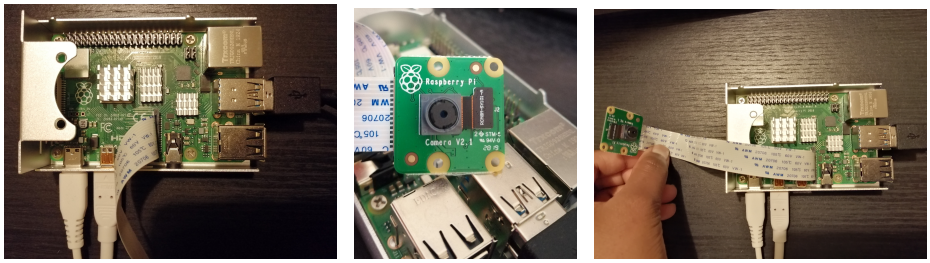

Fig. 2. A picture of the edge computing experiment setup used in this paper. Extreme-left is a Raspberry Pi 4 Module, center is a Pi camera module and the extreme-right image shows the connection of Pi camera on the Raspberry Pi Module. The USB ports helps connect the mouse and keyboard for control and the display device is connected to the HDMI port.

In this work on person detection, the input will be captured by a pin-point camera device which will be mounted on an edge processor. The real-time streaming and processing of image is necessary at the edge device to initiate the detection process on the edge device. The Edge computing device used in this work is a Raspberry Pi 4 with 4GB RAM. A picture of the experiment setup is attached above in figure 2.

In [10], Deng et.al. describes the principle of edge computing as the process of transferring the computation and communication resources to the edge of networks, from the cloud, to reduce the latency, enabling faster responses for end users. They also state that Edge Intelligence is a blooming field today. Studies shows that by 2024, 40-ZB of global internet data will be generated by IoT devices. In contrast to the growth rate of data generated by the IoT devices, the global datacenter traffic is estimated to reach only 20.6ZB by 2021 [25]. At this stage, the conventional wisdom is to transfer the data into cloud datacenters which will lead to network congestion. This is the reason for the recent advancements to handle user demands at the edge-cloud servers. The process of analyzing user data at the edge device directly can account for the concerns of latency, monetary loss in data transfer and privacy issues associated with data transfer and storage.

The large model size and complex matrix calculations during the inference process in this use case poses a significant challenge for the deployment of deep learning models on edge devices [26]. Re-design of deep learning architectures become inevitable given the increased need of efficient performance of deep learning frameworks on resource constrained devices [19].

3 Model Optimization Efforts

The efforts of model optimization can be classified into different types based on the change in the architecture. A significant property of Deep Neural Networks is that its inference is not affected by minor changes in weights or activation

functions. Hence the optimization techniques started off as two-fold: modification of network structure to increase efficiency, which led to MobileNets which use depth-wise separable convolutions, and the second category is introduction of quantization from floating point precision to discrete levels, owing to quantized inference due to the constrained weights and activation values [19].

Training large deep learning networks required computing clusters of thousands of machines and various methods like ensemble models were studied in the process of constructing smaller, compressed models. The model compression techniques were later classified into four in [7] based on the principles used. 1) Reduction in Size by pruning, quantization and model compression. 2) Altering the matrix multiplication by matrix factorization and filtering. 3) Based on domain knowledge and the data learned which includes processed like knowledge Distillation and Transfer Learning. 4)Hybrid methods.

3.1 Mimic Nets and Mimic Loss

Neural networks have undergone different stages of evolution and further size optimizing stages. Mimic architectures were one the first experiments, further progressing into various model compression techniques. Mimic nets are architectures which are not necessarily supporting the neurological analogy, but they are used to mimic consistent training data [16]. Glenn et.al introduced mimic nets as a technique to train feed-forward nets to automate classification and ranking task in 1993.

Mimic nets have two stages of operation. The first stage is to generate an augmented feature space from the training data input features and in the second stage, linear boundaries to separate classification categories, the selected and rejected options, are found. Feature selection is a substantial part of constructing a mimic net as it also defines the order of the mimic net.

Mimic nets could classify inputs as well as rank sets of input feature and have since been used to optimize classification and ranking tasks. Recently in 2018 [4], Plantinga et.al used a mimic architecture that was developed to mimic the output of a spectral classifier and called it mimic loss. Mimic loss mentioned by Plantinga is used to train a student model on a different task than the teacher model. This solution only applies to classification and ranking problems which make it out of scope in person detection problem.

With the increased use of analytics, Data Protection has also become seriously important in every domain, be it industrial or academic or personal. This concept of stealing data from Deep Neural Networks is possible with Mimic Nets which were initially developed to learn the generalization function learned by large deep learning networks. Mosafi et. al in 2019 [20] proposes the need for protecting the data learned by networks which are deployed at the core of AI based products and services. Though Mimic nets have proved to give promising results in classification problems, they are susceptible to the threat of data stealing, by mimicking the output of student network using a random large dataset. This makes Mimic net an unfavorable candidate for Edge Deployment.

3.2 Model Compression

Model Compression has been an area of research since large deep neural networks have been used for solving problems. Ensemble models were an option to use smaller models to solve problems faster, but the main disadvantage of ensemble models is that they are not suitable for applications in which real-time predictions are needed, or in case of portable devices and sensor networks [6]. In 2006, Bucila et.al [6] proposed the use of model compression to obtain fast, compact and accurate models.

Shallow feed-forward nets can learn deep functions using the same number of parameters as the original deep models. The concept that is experimented by Jimmy et. al [3], is that the shallow nets can be trained that perform similar to complex well-organized deeper convolutional models. In the work on convolutional nets by Urban et. al [23], the main emphasis was on analysis of CIFAR-10 Dataset which gave poor results when the shallow nets were trained to learn the function from the deep learning networks and the presence of convolutional layer was inevitable. Convolutional layers have proved to be the best neural network layers for extracting feature information from image data and hence is important to our problem of person detection from images and live video stream. Depth in neural networks improve generalization capability of the model. Hence the most effective method of model compression adopted lately is model quantization [15].

Hinton and Vinyals, a team from Google, found that an effective way to transfer the generalization capability from a large model or ensemble of models is to use a single model [13]. This process of transfer of the large cumbersome model to a small model using a different kind of training is called Knowledge Distillation. Hinton et. al [13] trained models with no convolutional layers on CIFAR-10 dataset with an accuracy of 70.2% using distillation. In knowledge distillation, knowledge can be transferred from one model to another model with different architecture by training the new model on a transfer set.

In knowledge transfer learning, a base(teacher) network is first trained and then re-purpose the network. In the re-purposing step, the knowledge is transferred to a second target(student) network to be trained on a random target dataset. This kind of transfer learning has been found to give promising results for learning on edge devices [25].

4 Quantization Effects

Network quantization achieves large reduction in memory and processing power usage by reduction in precision values and operations within a model [1]. Quantization enables on-device inference for deep learning models residing on Edge Devices. Person detection can be achieved in Edge Devices using quantized Deep Learning models. This work compares the accuracy of detection and inference time required by the non-quantized model vs quantized models. Unlike shallow net, the network architecture is not altered in quantization process and the network remains deep enough for better generalization capability.

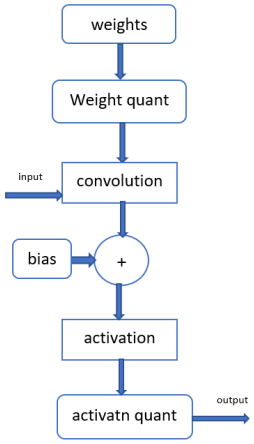

Fig. 3. The flow-diagram of training with simulated quantization [15]

In the flow-diagram, the *'weight quant'* and *'activatn quant'* are quantization nodes integrated into the computation graph to simulate the effects of quantization of the respective weight and activation values. Training that accounts quantization in models accounts for quantization error during training to reflect the quantization at the point of inference. In this type of quantization, every quanti-zation is followed by dequantization, thus facilitating the simulation of precision loss in case of inference operation using arithmetic operations. Such quantization is termed as quantization-aware training. In Post training quantization, the inference is quantized with offline conversion from floating point to fixed point.

In Post training Quantization, model size can be reduced by quantizing an already trained float TensorFlow model. There are three types of quantization available. They are 1)Dynamic Range Quantization which statically quantizes only weights from floating point to integer(with 8-bits precision) which are then converted back to floating point during inference. In this process, the model can become 4times smaller, with 2-3 times increase in speed. 2)Full integer Quantization, as the name suggests, all mathematical operations are integerised and hence 3times increase in speed and reduction in peak memory usage. This is mostly used in Edge devices. 3)Float16 quantization quantizes the weight to float16 from floating point numbers. This ensures minimal loss in accuracy and mostly used only with CPU and GPU devices. The quantization method adapted in this experiment is dynamic range quantization, which is a type of post-training quantization.

Quantization aware training is better for model accuracy than post training quantization, even though the latter is often easier to achieve [2]. The quantized models use 8-bit float instead of 32-bit float and this is more similar to inference-time quantization.

5 Experiment and Analysis

The experimentation were carried out by training the models using the TensorFlow object detection API and with the use of transfer learning, the models are then trained and fine-tuned for person-detection task. TensorFlow Lite is the framework for inference modules to work on resource constrained devices with low latency. The weights are converted into 8-bit precision values in the TensorFlow Flat buffer format. The activations are always stored in floating point. For inference, the activations are dynamically quantized to 8-bit precisions prior to inference processing and dequantized back into floating point post processing.

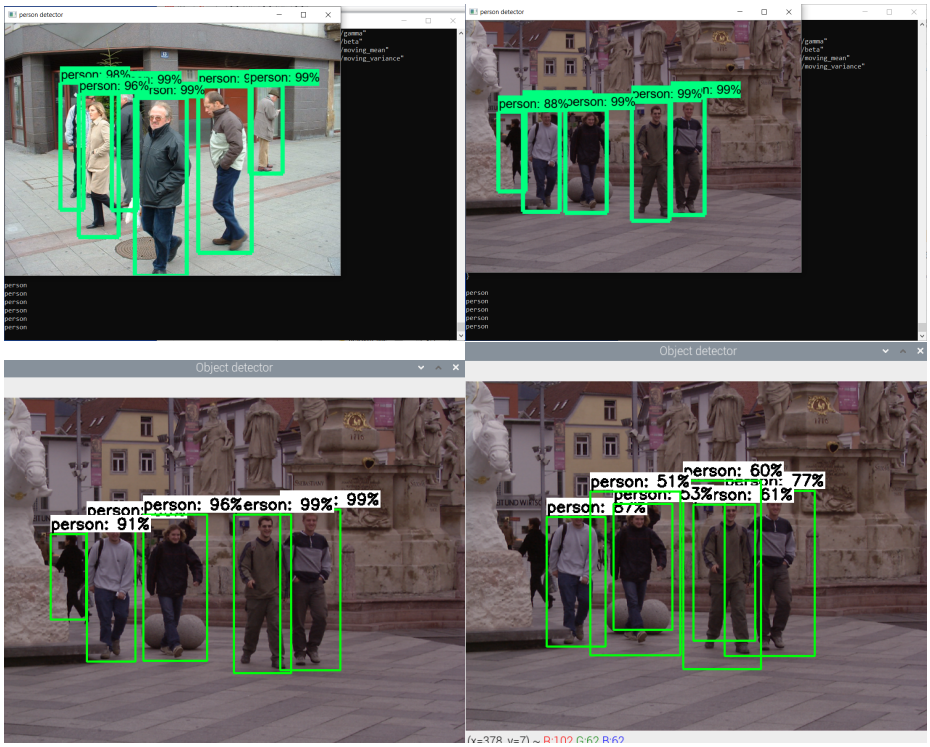

Fig. 4. The top two images show person detection from the INRIA dataset using unquantized SSD MobileNet Model. The bottom row of pictures show detections from the Raspberry Pi. The bottom-left image is a fully accurate detection comparable to the GPU and the bottom-right is an image with two False-Positives

The reference experiment for the model optimization criteria is SSD inception network trained on INRIA dataset [9]. This is included as the first model in both the comparison tables. In 2014, Szegedy et.al from Google proposed GoogLeNet(22 layers) which consists of inception modules and hence came to

Table 1. Comparison of different SSD models on the GPU vs the Raspberry Pi in person detection based on detection time and IOU value

Object detection Model Name	Time for detection(ms)	IOU value
SSD-Inception-v2-coco(GPU)	3.247	0.8034
SSD-Mobilenet-v1-coco(GPU)	2.119	0.7699
SSD-Mobilenet-v2-coco(GPU)	1.811	0.713
SSDLite-Mobilenet-v2-coco(GPU)	1.332	0.6337
SSD-Inception-v2-coco(Rasp-Pi)	2.309	0.7081
SSD-Mobilenet-v1-coco(Rasp-Pi)	1.027	0.7829
SSD-Mobilenet-v2-coco(Rasp-Pi)	0.813	0.7378
SSDLite-Mobilenet-v2-coco(Rasp-Pi)	0.648	0.591

be widely known as Inception Net [21]. These networks were further modified in architecture in 2015, which led to versions Inception-v2 and Inception-v3 [22]. MobileNets were lighter networks, also designed by Google engineers for Mobile vision applications [14]. The models were trained on TensorFlow Model repository on the TensorFlow version 1.14. The inference was programmed using Python 3.6 with OpenCV 3.4 as the image analysis package. The training process is completed using GPU GeForce RTX 2080 Ti and the frozen graph is exported as a TensorFlow-lite model. This model is then converted into a flat-buffer format used for detection experiments on Raspberry Pi 4 running the Raspbian Buster-10 OS. The Raspberry Pi Camera Module V2 is used for real-time detection experiment. The simulation in this experiment involves detection on images for controlled comparison of results. The Raspberry Pi camera is single channel, 8megapixels and has a maximum frame rate capture of 30fps. To connect the camera module to a pi, a 15cm ribbon cable is attached to the module slots into the Pi Camera Serial Interface port(CSI).

Table 2. Comparison of different SSD models on the GPU vs the Raspberry Pi in person detection based on Precision and Recall value of detection

Object detection Model Name	Precision	Recall
SSD-Inception-v2-coco(GPU)	1.0	0.8
SSD-Mobilenet-v1-coco(GPU)	1.0	1.0
SSD-Mobilenet-v2-coco(GPU)	1.0	0.8
SSDLite-Mobilenet-v2-coco(GPU)	0.8	0.8
SSD-Inception-v2-coco(Rasp-Pi)	1.0	1.0
SSD-Mobilenet-v1-coco(Rasp-Pi)	1.0	1.0
SSD-Mobilenet-v2-coco(Rasp-Pi)	1.0	0.8
SSDLite-Mobilenet-v2-coco(Rasp-Pi)	0.66	0.8

The evaluation metric used in this experiment is IOU(Intersection Over Union). The IOU is the ratio of area of intersection vs area of union of the predicted bounding box and the ground-truth bounding box. This is used to

measure the Precision and Recall of object detection. Precision is the ratio of True Positives against all positive detections whereas Recall is the ratio of True Positives against the sum of True Positives and False Negatives(all ground truth instances).

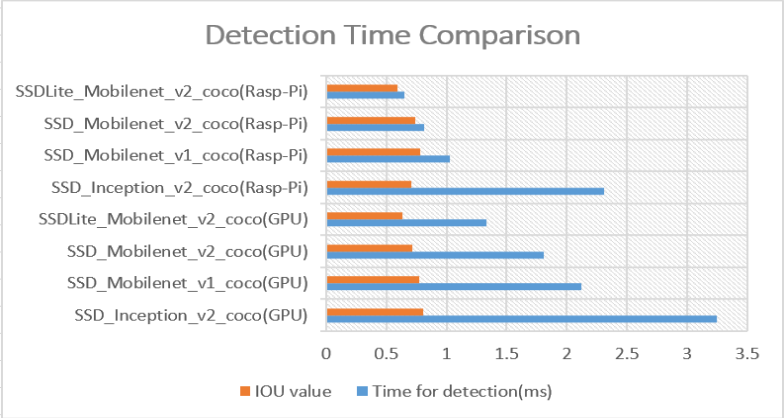

Fig. 5. The graphical comparison of detection time vs IOU value for all the models used. The graph clearly shows comparable detection time(in ms)

The above graph shows that the inference time of the TensorFlow Lite model is comparable to the larger models with slight change in IOU values. This is a promising result. In case of industrial application, accuracy is a concern and a 10% reduction in accuracy might lead to much higher number of erroneous products and which might cost huge money in any mass production system. If the acceptable floor rate of detection in an industrial scenario is considered to be above 70% and within the time of detection of 3ms, then quantized version of SSD-Inception-v2 and SSD-MobileNet are winning candidates. SSD-MobileNet-v1 is a framework designed for the mobile and edge devices and it has been found to be the most accurate and robust in the person detection experiment with a detection rate of 78% and detection time of 1ms. The SSDLite-Mobilenet model does not perform well enough to be considered for any industrial application as the IOU values falls to the value of 59% and very low precision of 0.66. The floor rate of detections as above 70% is sufficient for reliable identification of person which is a standard engineering performance requirement for a cell-based manufacturing environment.

The research for model optimization is still ongoing with different types of quantization under study. Further research in this area of model quantization considers adaptive quantization and layer-wise quantization. The process of quantization is the most-effective method of model compression used in the current research for autonomous person detection on Edge Devices.

6 Conclusion

Edge Intelligence is rapidly developing with optimization of Deep Neural Networks. Among all the types of optimization and compression techniques, Model quantization is the most widely used optimization method, due to its huge reduction in size as well as reduction in computational costs. Further developments in model quantization like adaptive quantization or multiple quantization within the same neural network model are also under research. The Knowledge Transfer methods discussed above have been found beneficial in classification problems but in case of person detection, model quantization gives the best results with reduced inference time. The quantized models work efficiently on the Raspberry Pi module, the edge device, with high accuracy values of more than 70% with a reduced detection time of less than 3ms, which is comparable to the detection time in GPU accelerated devices with non-quantized models. Thus, person detection in Industrial application can rely on quantized models for stand-alone Edge Devices. This highly accurate detections can be further integrated into intelligent automated responses

References

1. Get started with tensorflow lite, https://www.tensorflow.org/lite/guide/get_started, [Online; accessed 11-October-2020]
2. Post training quantization in tensorflow lite (2020), https://www.tensorflow.org/lite/performance/post_training_quantization, [Online; accessed 11-October-2020]
3. Ba, J., Caruana, R.: Do deep nets really need to be deep? In: Advances in neural information processing systems. pp. 2654–2662 (2014)
4. Bagchi, D., Plantinga, P., Stiff, A., Fosler-Lussier, E.: Spectral feature mapping with mimic loss for robust speech recognition. In: 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 5609–5613. IEEE (2018)
5. Braun, M., Krebs, S., Flohr, F., Gavrila, D.M.: Eurocity persons: A novel benchmark for person detection in traffic scenes. IEEE transactions on pattern analysis and machine intelligence **41**(8), 1844–1861 (2019)
6. Buciluǎ, C., Caruana, R., Niculescu-Mizil, A.: Model compression. In: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 535–541 (2006)
7. Chen, C.J., Chen, K.C., Martin-Kuo, M.c.: Acceleration of neural network model execution on embedded systems. In: 2018 International Symposium on VLSI Design, Automation and Test (VLSI-DAT). pp. 1–3. IEEE (2018)
8. Chollet, F.: Deep Learning mit Python und Keras: Das Praxis-Handbuch vom Entwickler der Keras-Bibliothek. MITP-Verlags GmbH & Co. KG (2018)
9. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: 2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05). vol. 1, pp. 886–893. IEEE (2005)
10. Deng, S., Zhao, H., Fang, W., Yin, J., Dustdar, S., Zomaya, A.Y.: Edge intelligence: the confluence of edge computing and artificial intelligence. IEEE Internet of Things Journal (2020)

11. Engel, J., Hantrakul, L., Gu, C., Roberts, A.: Ddsp: Differentiable digital signal processing. arXiv preprint arXiv:2001.04643 (2020)
12. George, D., Huerta, E.: Deep learning for real-time gravitational wave detection and parameter estimation: Results with advanced ligo data. Physics Letters B **778**, 64–70 (2018)
13. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531 (2015)
14. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861 (2017)
15. Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., Kalenichenko, D.: Quantization and training of neural networks for efficient integer-arithmetic-only inference. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2704–2713 (2018)
16. Johnson, G.E.: Mimic nets. IEEE transactions on neural networks **4**(5), 803–815 (1993)
17. Knezović, J., Pervan, B., Relja, Z., Knezović, J.: Project houseleek-a case study of applied object recognition models in internet of things (2019)
18. Kristiani, E., Yang, C.T., Huang, C.Y.: isec: An optimized deep learning model for image classification on edge computing. IEEE Access **8**, 27267–27276 (2020)
19. Kwasniewska, A., Szankin, M., Ozga, M., Wolfe, J., Das, A., Zajac, A., Ruminski, J., Rad, P.: Deep learning optimization for edge devices: Analysis of training quantization parameters. In: IECON 2019-45th Annual Conference of the IEEE Industrial Electronics Society. vol. 1, pp. 96–101. IEEE (2019)
20. Mosafi, I., David, E.O., Netanyahu, N.S.: Stealing knowledge from protected deep neural networks using composite unlabeled data. In: 2019 International Joint Conference on Neural Networks (IJCNN). pp. 1–8. IEEE (2019)
21. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2015)
22. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2016)
23. Urban, G., Geras, K.J., Kahou, S.E., Wang, O.A.S., Caruana, R., Mohamed, A., Philipose, M., Richardson, M.: Do deep convolutional nets really need to be deep (or even convolutional)? (2016)
24. Verstraete, D., Ferrada, A., Droguett, E.L., Meruane, V., Modarres, M.: Deep learning enabled fault diagnosis using time-frequency image analysis of rolling element bearings. Shock and Vibration **2017** (2017)
25. Zhou, Z., Chen, X., Li, E., Zeng, L., Luo, K., Zhang, J.: Edge intelligence: Paving the last mile of artificial intelligence with edge computing. Proceedings of the IEEE **107**(8), 1738–1762 (2019)
26. Zhu, X., Zhou, W., Li, H.: Adaptive layerwise quantization for deep neural network compression. In: 2018 IEEE International Conference on Multimedia and Expo (ICME). pp. 1–6. IEEE (2018)