Real-Time Embedded Intelligence System: Emotion Recognition on Raspberry Pi with Intel NCS

Y Xing*[1], P Kirkland*[1], G Di Caterina[1] J Soraghan[1] G Matich[2]
*contributed equally to first author

[1] University of Strathclyde, Glasgow, UK [2] Leonardo MW, UK

{ yannan.xing; paul.kirkland; gaetano.di-caterina;j.soraghan}@strath.ac.uk

Abstract. Convolutional Neural Networks (CNNs) have exhibited certain human-like performance on computer vision related tasks. Over the past few years since they have outperformed conventional algorithms in a range of image processing problems. However, to utilise a CNN model with millions of free parameters on a source limited embedded system is a challenging problem. The Intel Neural Compute Stick (NCS) provides a possible route for running large-scale neural networks on a low cost, low power, portable unit. In this paper, we propose a CNN based Raspberry Pi system that can run a pre-trained inference model in real time with an average power consumption of 6.2W. The Intel Movidius NCS, which avoids requirements of expensive processing units e.g. GPU, FPGA. The system is demonstrated using a facial image-based emotion recogniser. A fine-tuned CNN model is designed and trained to perform inference on each captured frame within the processing modules of NCS.

Keywords: CNN, Embedded System, Low Power System, SWAP profile

1 Introduction

Size, Weight and Power (SWaP) profile are important factors in many applications of real-time embedded systems. However, it is difficult to incorporate the benefits of deep learning (DL) in real-time embedded systems due to the limited computation capability and power. One solution is to use cloud computing [1]. This paper shows the first comparison between typical DL hardware and an edge device, which is applicable to any DL model that does not require any online learning. Hoping to show how the NCS can help bridge the gap between the two. The next phase of the Internet of Things (IoT) development will be adding intelligence to the devices. This will not only allow each device to share more in-depth information, but it will also require less information to be sent off the device which provides a greater level of security. These devices are typically unable to run DL models due to the amount of processing required, which we show is no longer the case.

The remainder of the paper is organised as follows. Section 2 introduces the background of DL on embedded systems. Section 3 describes the proposed Ras-Pi NCS system with details on system configuration working with a very simple self-designed CNN based on public emotion recognition dataset. The runtime speed and power consumption of system are presented in Section 4and compared to several DL evaluation platforms. Section 5 concludes the paper.

2 Background

Advances in Smart devices and Internet of Things have led to a plethora of devices with the potential to have real-time embedded intelligence. As traditional DL research concentrated on GPUs, the focus was on computational runtime, accuracy and not power use [2]. However, low powered edge devices are an important area of research.

Nvidia to date has arguably the most popular embedded DL devices with the Jetson range. With the NVidia Jetson range TK1, TX1 and TX2 have featured in up to 1,000 research publications (Google Scholar Search) in a broad range of applications [3-5]. Benchmarks compared to PCs have shown there are good use cases for these devices, with power savings of 15-30 times, with a reduction in throughput by one-tenth, a net increase of 5-15 times [6].

Other chip manufacturers have introduced their own version of low powered embedded DL accelerators, such as Qualcomm with their Snapdragon Neural Processing Engine and Intel with both their Nervana Neural Network Processor and the device featured in this paper, the Movidius Myriad Visual Processing Unit (in the form of the Neural Compute Stick). Pena et.al [7] presented the first benchmarking results of the NCS, Raspberry Pi 3, and the Intel Jolue 750x development board. All tests are measuring the amount of time taking and power used to complete one pass of the network, with the NCSs results being averaged over both boards. It also showed how the different systems handled a variety of differing complexity networks. Figure 1 gives an illustration of the design system. The NCS, Webcam and Power Bank are connected via USB, while the touchscreen is connected via the GPIO pins to the Raspberry Pi. The actual system showing how all the components are connected and how the UI appears on the screen are shown on the right of Fig. 1.

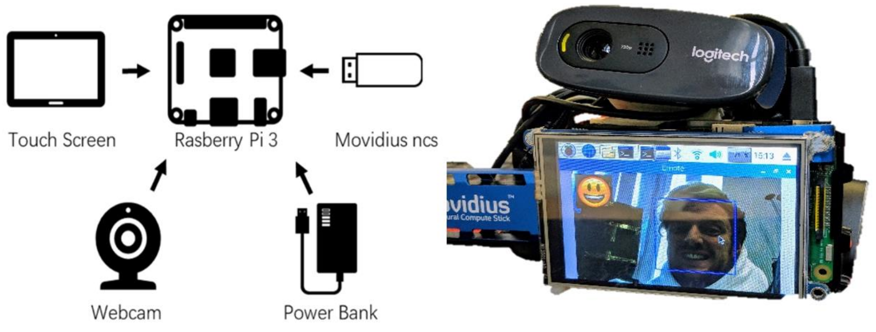

Fig. 1. The Raspberry Pi – NCS System.

3 The System

As is shown in Fig.1 a Raspberry Pi 3 model B development board with 40 GPIO pins and four USB2.0 ports is used as the main processing unit in the proposed system. The power of board is supplied by a 5V/2A mobile DC power bank to make the whole system fully portable. The Raspberry Pi contains a 1.2 GHz 64-bit quad-core ARM Cortex-A53 CPU with 1GB of RAM, running standard RASPBIAN Jessie desktop OS that supports required Python programming environment for the Intel NCS SDK to run in API mode. A Logitech C270 HD (720p) webcam is used via one USB2.0 port to provide desired video input. A 3.5” TFT touchscreen is set to satisfy general user interaction and visualising the online-processing results. The Intel NCS is used exclusively for the neural network model, with the information being transferred over the USB interface.

3.1 The CNN for Emotes

The architecture of the network had to be a reduced and cut down version of the state of the art architectures, as the inference models that are runnable on the NCS cannot resolve the unknown placeholders/variables. Very often these placeholders are employed for training specific parameters but are not necessary for NCS inferencing. Before trying to compile the NCS model, the Tensorflow model can be trained to generate three saved model files: index file as model indexing, data file as the network parameters and meta file which contains the network structure. These files will be further used in the shrunken version of the network to generate another set of inference-only network models. The original network is reduced with dropout layers and training specific code removed which usually contains: reading/importing data, loss function and accuracy computation definition, placeholders except for the input tensor of the network etc. The name for the input and output layer always requires to be set to make sure the compiler is easier to determine and recognise from the structure of the network.

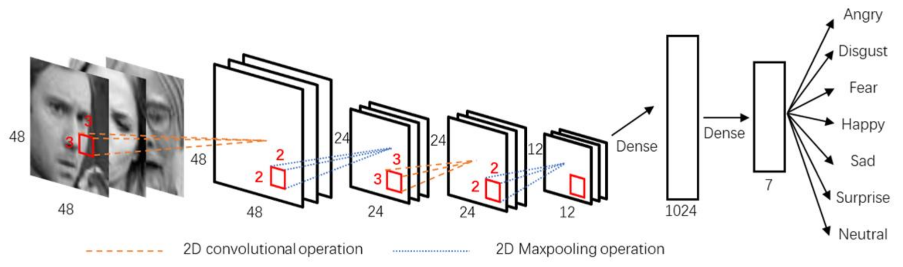

Fig. 2. The designed CNN architecture for facial expression recognition, the stride of convolution kernel is set to be 1 and the stride of polling kernel is set to be 2.

The facial based emotional recogniser network illustrated in Fig.2 contains a total of 6 layers: 2 convolutional, 2 pooling, and 2 fully-connected. The Rectified linear unit (ReLU) activation function [8] is employed for each layer. The CNN inputs are grayscale 48x48 images and followed by 2 convolutional layers with 32 and 64 filters with the size of both being 3x3. The convolution operation in each layer is set to be 1-pixel strides with the same padding. The max-pooling layers with 2x2 kernel size is placed behind each convolutional layer to perform a subsampling for feature maps from the previous layer. The final stage is a fully connected dense layer with 1024 neurons, the network output layer is comprised of 7 neurons performing the softmax[9] calculation which indicates the number of facial emotions.

Training phase. The training data utilises the FER2013 database [10]. According to the results from previous work, we consider that the order of the original dataset represents an unbalanced training set., which can lead to obvious overfitting and underfitting issues. We randomly initialised the order of the originally given dataset as well as correspond labels and split it into required data batches. The learning of the CNN employed the backpropagation incorporating cross-entropy as target loss function and the Adam stochastic optimiser [11]. In order to prevent the network suffering from overfitting, the Ridge Penalisation (L2 regularisation) is implemented among the cross-entropy function. The dropout technique [12] also well known as an effective regularisation to prevent network overfitting. In the training phase, the fully connected layer is set to randomly dropout with rate 0.5.

Result. The designed convolutional network was validated on the self-defined (shuffled) testing set and validation set. The Extended Cohn-Kanade(CK+) [13] dataset was also used to evaluate the actual performance of the network with the same model which trained by the FER2013 dataset. After 100 epochs the model converged at 90.99% testing accuracy and 87.73% validation accuracy based on the shuffled FER2013 dataset. The model showing a 70.51% test accuracy on the CK+ dataset with a very good performance in recognising happy with 100%, supervised with 92% and neutral with 84%. The confusion matrix on the FER2013 test set shows nearly perfect accuracy on each emotion.

3.2 Embedded Device

The Raspberry Pi 3 was the chosen flexible platform for this work. As a Linux microcomputer, it can run a multitude of programs similar to a Desktop PC. In our problem, it needed to be able to run the Tensorflow deep learning environment and the full Intel NCS SDK (although we are only going to utilise and use the API to save on space). This approach makes use of the Raspbian stretch desktop OS while utilising another 18 libraries. Along with API, the emotion recognition system makes significant use of two other important libraries, OpenCV [14], an open source computer vision library that is used for the Haar cascade face detection function and to display the live emotion recognition feed to the display. The decision to use the inbuilt Haar cascade was due to the speed at runtime compared to similar models, once optimised, for example, the dlib [15] libraries Histogram of Gradients (HoG) face detector. The Haar classifier was able to produce a robust detection of a face at 3-4x the speed of the HoG classifier. The Haar classifier is known to be the faster and less accurate classifier but proves to be robust enough as the face cropper within this system. The well-established Multi-Task Cascaded Convolutional Network (MTCNN) [16] deep learning approach to face detection and alignment was also tested, as a recent implementation had appeared on the NCS GitHub page called the NCS App Zoo. This network requires the use of a further 2 Intel NCS units to run but would allow an end-to-end deep learning approach to the emotion recognition. However, due to the bounding box regression stages of the network, it proved to be slower with an average runtime similar to that of the dlib HoG classifier.

The other important library to ensure real-time operation is imutils [17]. This convenience library helps with a significant speed up with one of the bottleneck areas of image acquisition. Compared to the OpenCV function, the imutils function utilises the multi-threading of the Raspberry Pi's quad-core processor having a function able to collect the image from the webcam as soon as it is available and then store it to a queue of images. The result is the time taken to acquire the next image reduces from 10s of milliseconds to 10s of nanoseconds.

3.3 Intel NCS

DL is appearing in an increasing number of mobile devices without the necessity for cloud computing. The NCS used in this paper is from chipmaker Intel's Movidius department, which incorporates one Myriad 2 machine vision processing unit into a small USB stick, Movidius announced that it delivers more than 100 gigaflops of performance. It can locally run neural networks inference model using Caffe and Tensorflow framework. A general development process of NCS based embedded system is illustrated in Fig 3. The training process does not need to utilise the NCS stick or SDK but only standard DNN development on a desktop PC. Using the software SDK of the NCS, the user should subsequently perform training, profiling, tuning and compiling a DNN model on the NCS and a PC that runs x86 64bit ubuntu 16.04 OS. The provided SDK can check the validity of designed DNN and API for python and C languages. After that, any developer system (e.g. a raspberry pi) that runs a compatible OS with neural compute API can accelerate neural network inferences.

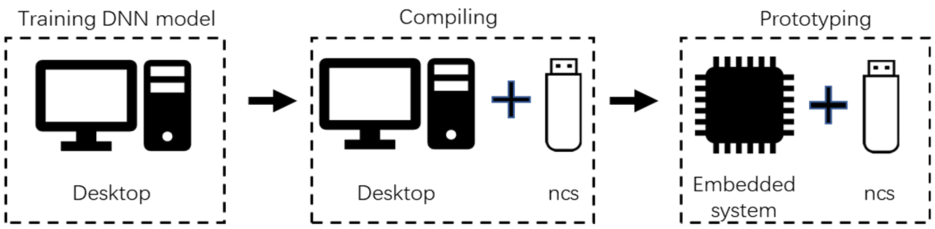

Fig. 3. Illustration of using the Intel NCS to develop for a DNN based embedded system

4 Evaluation

This section looks at the running of the emotion recognition program and delivers the results in terms of processing time. The Raspberry Pi 3 is compared against two other devices running the same application both running Ubuntu 16.04. An Alienware 15 Laptop with an Intel i7 6820HK Quad-core CPU @ 2.7 GHz with 16GB of RAM and a GTX 980m GPU. The other machine was a Desktop PC with an AMD Ryzen 1700X Octa-core CPU @ 3.4GHZ with 32GB of RAM and a GTX 1080Ti., with the added measure of how much power is being used to deliver the results.

4.1 Real Time Running on Pi

The actual system with all devices is illustrated in Fig.1. Fig. 4 shows runtimes, for 10 runs of the code averaged, each running for 300 frames. The times for the non-deep learning parts are combined and averaged so as not to influence the overall results as only minor fluctuations appear between runs. Fig. 4, breaks down the timing into sections:

- *Camera Read* – the time taken for reading an image from the camera
- *Image Show* – the time taken to display the image onto the touchscreen with emotion emoji,
- *Haar Face Detection*-the time taken for the detector to crop the image around a found face
- *Inference Runtime* – the time taken for the CNN model to run and
- *Loop Runtime*- the total time taken to process one image of the video capture.

The graph highlights which parts of the process are slow, especially on an embedded device, with the time axis given in a log format to allow for the differing magnitudes of time taken for different tasks to be represented equally. Only one section differs from the previously mentioned arrangement which is the R-Pi (opt) which is the optimised version to allow the system to run in true real-time (sub 33ms), while still displaying the camera feed to the user. Modifications included running the Haar face detector only every 3rd frame. Also removing the emoji image to the image displayed on the device and instead, printing the emotion to the terminal. This resulted in a saving of 20 and 12 milliseconds respectively. Therefore, while the Pi with NCS can run at 14fps (66ms) the optimised code can run at 30fps (33ms), both of which can be classed as real-time, though the latter is obviously the preferred to perceive smooth motion on the video feed. Meanwhile, the Laptop and PC can output 142-167 fps (7 and 6ms respectively), though to do so they consume a considerable amount more power which is typically an undesired trait for an embedded device.

4.2 Benchmarking

Fig. 4, shows a significant speed up for the other processes in the application, with the GPUs managing to run the TensorFlow models with the fastest time as expected. A better comparison though is to see how the systems perform in terms of Inference per Second per Watt, which would be an important factor to consider with a minimal SWaP profile. Table 1 shows how the results for the 6 system types, with the new value given the term RP (Run-Power is the coefficient - inference/second/Watt).

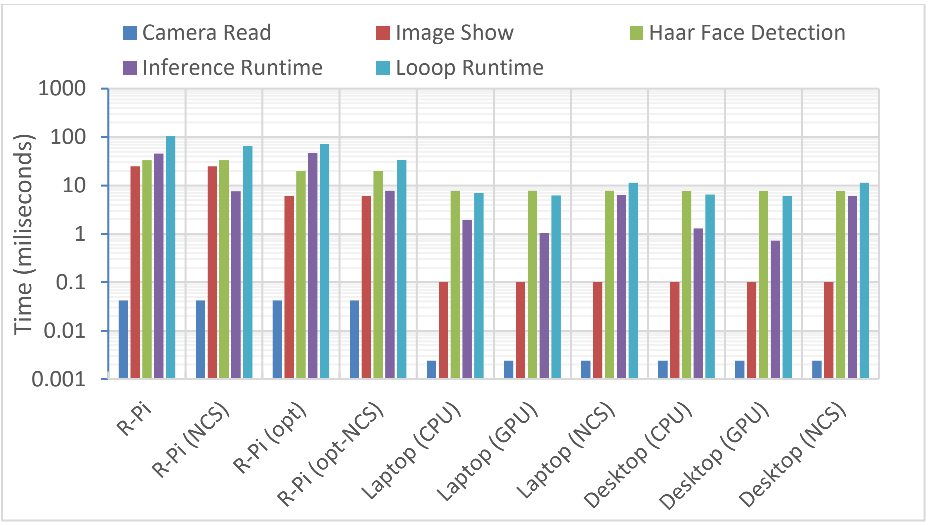

Fig. 4. Chart of Results showing each device's running time for the given section

Table 1. Results of power related inferences.

System	Time (ms)	Power (watt)	RP (inf/s/W)
R-Pi (+ NCS)	45.26 (7.57)	5 (6.2)	4.42 (21.31)
Intel CPU (+ GPU)	1.92 (1.04)	45 (167)	11.57 (5.76)
AMD CPU (+ GPU)	1.30 (0.73)	95 (345)	8.10 (3.97)

The results of the final experiment show that the Intel NCS given its low power usage of 1.2W, rates highly when given these SWaP constraints. Which vary per application in many of the DL use cases of Robotics, Human-Computer Interaction, Healthcare Application and several other Autonomous Systems. Given size and weight or even cost as extra parameters, the NCS would perform even higher than shown.

5 Conclusion

This paper presented a novel design concept, that shows how the Intel NCS device can help to bring state of the art DL to low powered edge devices. The combination of Raspberry Pi and NCS demonstrated the potential of these devices to help carry out complex image processing in real time similar to the Nvidia Jetson, the Intel NCS can be applied to almost any DL research area. This shows the ability of this low-cost inference model runner, to bridge the gap between current edge devices and desktop PCs for DL applications. With growing research into low powered embedded intelligence devices, this paper highlights the usefulness of this type of device.

References

1. Announcing Amazon Elastic Compute Cloud (Amazon EC2) - beta". Amazon.com. 24 August 2006
2. Shi, S., Wang, Q., Xu, P., Chu, X.: Benchmarking State-of-the-Art Deep Learning Software Tools, https://arxiv.org/abs/1608.07249.
3. Tomè, D., Monti, F., Baroffio, L., Bondi, L., Tagliasacchi, M., Tubaro, S.: Deep Convolutional Neural Networks for pedestrian detection. Signal Processing: Image Communication. 47, 482-489 (2016).
4. Paszke, A., Chaurasia, A., Kim, S., Culurciello, E.: ENet: A Deep Neural Network Architecture for Real-Time Semantic Segmentation, https://arxiv.org/abs/1606.02147.
5. Smolyanskiy, N., Kamenev, A., Smith, J., Birchfield, S.: Toward Low-Flying Autonomous MAV Trail Navigation using Deep Neural Networks for Environmental Awareness, https://arxiv.org/abs/1705.02550.
6. Rungsuptaweekoon, K., Visoottiviseth, V., & Takano, R. Evaluating the power efficiency of deep learning inference on embedded GPU systems. 2017 2nd International Conference on Information Technology (INCIT) (pp. 1-5). IEEE. (2017).
7. Pena, D., Forembski, A., Xu, X., Moloney, D.: Benchmarking of CNNs for Low-Cost, Low-Power Robotics Applications. RSS 2017 Workshop: New Frontier for Deep Learning in Robotics, 2017
8. Nair, V., & Hinton, G. E. Rectified linear units improve restricted boltzmann machines. In Proceedings of the 27th international conference on machine learning (ICML-10) (2010). (pp. 807-814).
9. Nasrabadi, N. M. (2007). Pattern recognition and machine learning. Journal of electronic imaging, 16(4), 049901.
10. Challenges in Representation Learning: A report on three machine learning contests." I Goodfellow, D Erhan, PL Carrier, A Courville, M Mirza, B Hamner, W Cukierski, Y Tang, DH Lee, Y Zhou, C Ramaiah, F Feng, R Li,X Wang, D Athanasakis, J Shawe-Taylor, M Milakov, J Park, R Ionescu,M Popescu, C Grozea, J Bergstra, J Xie, L Romaszko, B Xu, Z Chuang, and Y. Bengio. arXiv 2013.
11. Kingma, D., Ba, J.: Adam: A Method for Stochastic Optimization, https://arxiv.org/abs/1412.6980.
12. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research, (2014). 15(1), 1929-1958.
13. Lucey, P., Cohn, J., Kanade, T., Saragih, J., Ambadar, Z., Matthews, I.: The Extended Cohn-Kanade Dataset (CK+): A complete dataset for action unit and emotion-specified expression. 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops. (2010)
14. Bradski, G., & Kaehler, A. OpenCV. Dr. Dobb’s journal of software tools, 3. (2000).
15. King, D. E. Dlib-ml: A machine learning toolkit. Journal of Machine Learning Research, 10(Jul), (2009).1755-1758.
16. Zhang, K. Zhang, Z., Li, Z., & Qiao, Y. Joint face detection and alignment using multitask cascaded convolutional networks. IEEE Signal Processing Letters, (2016). 23(10), 1499-1503.
17. Imutils Library https://github.com/jrosebr1/imutils 2018/05/27