Benchmarking of CNNs for Low-Cost, Low-Power Robotics Applications

Dexmont Pena, Andrew Forembski, Xiaofan Xu, David Moloney

Abstract—This article presents the first known benchmark of Convolutional Neural Networks (CNN) with a focus on inference time and power consumption. This benchmark is important for low-cost and low-power robots running on batteries where it is required to obtain good performance at the minimal power consumption. The CNN are benchmarked either on a Raspberry Pi, Intel Joule 570X or on a USB Neural Compute Stick (NCS) based on the Movidius MA2450 Vision Processing Unit (VPU). Inference performance is evaluated for TensorFlow and Caffe running either natively on Raspberry Pi or Intel Joule and compared against hardware acceleration obtained by using the NCS. The use of the NCS demonstrates up to 4x faster inference time for some of the networks while keeping a reduced power consumption. For robotics applications this translates into lower latency and smoother control of a robot with longer battery life and corresponding autonomy. As an application example, a low-cost robot is used to follow a target based on the inference obtained from a CNN.

I. INTRODUCTION

Recent advances in machine learning have been shown to increase the ability of machines to classify and locate objects using Convolutional Neural Networks (CNN). For robotics applications this translates into increased reliability for tasks like tracking and following a target. However these networks typically have a large number of parameters resulting in a large number of operations.

In order to perform the large number of operations required by CNNs in real time, current systems typically use setups including expensive GPUs or distributed systems. This limits the application of CNNs on low-cost robots typically used by students and hobbyists.

Due to the popularity of CNNs in recent years a variety of different frameworks have been proposed, these include CuDNN (limited to Nvidia GPUs) [13], Caffe [14], MXNet [12] and Tiny-DNN [6] and more recently Googles TensorFlow [7] which is currently the most popular deep learning framework. Some of these frameworks have been shown to work on embedded platforms like Raspberry Pi [3] and the Intel Joule [2] but due to hardware limitations processing can be slow, taking a few seconds per frame even for custom designed networks [25].

One alternative to circumvent the hardware limitations of low-cost robots is to use a dedicated hardware accelerator like the Neural Compute Stick (NCS) [1]. This USB device offloads compute-intensive CNNs while still being low-cost and low-power. In this paper we benchmark different Convolutional Neural Networks (CNN) for image classification focusing on power consumption and inference times on embedded platforms. Then we show an application of a low-cost follower robot.

II. RELATED WORK

Current benchmarks for Deep Learning Software tools limit the scope of their usage to the use of high power CPUs and GPUs and focus on the training time and accuracy [22]. No benchmarks have been found taking into account power consumption and running time on embedded systems. The only available benchmark focused on embedded systems evaluates only one network with a few layers[20] and provides no power consumption data.

Due to the processing power limitations of embedded platforms only a few alternatives exist that are able to perform inference on these low-power platforms. TensorFlow [7], Caffe [14] and MXNet [12] have been shown to compile on the Raspberry Pi [5] [20]. Tiny-DNN [6] was developed for low-resources in mind and is a header only library. Common libraries for image processing like DLib [15] and OpenCV [10] have been extended with modules to support DNNs however the number of networks they support is limited.

Only a few frameworks have been optimized for their use in low-cost low-power systems. Peter Warden optimized the DeepBelief SDK for the Raspberry Pi [25] however the current version requires three seconds to process a single frame using AlexNet [8]. Other libraries like OpenBLAS [24] focus on the Linear Algebra routines and can be used by some of the DNN frameworks to speed-up the processing. Caffe, Torch and MXNet include support for OpenBLAS to accelerate processing.

One of the main challenges for using the existing frameworks is the lack of standardization for storing the network
TABLE I: Software versions used for the evaluation of the DNN. TensorFlow is compiled with the flags ".\-mfpu\=neon\-vfpv4 \-funsafe\-math\-optimizations \-free\-vectorize".

<table>
<thead>
<tr>
<th>Software</th>
<th>GitHub Commit ID / Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenBlas</td>
<td>92085a7</td>
</tr>
<tr>
<td>Caffe</td>
<td>f731bc4</td>
</tr>
<tr>
<td>TensorFlow</td>
<td>1.0.1</td>
</tr>
<tr>
<td>NCS</td>
<td>v1.02</td>
</tr>
</tbody>
</table>

Although some of the frameworks offer parsers for Caffe \[^6\], \[^10\], \[^6\], \[^12\] the support for importing different network architectures is limited. The SDK for the NCS provides parsers for TensorFlow and Caffe, although with some limitations on the network architectures. Other alternatives for embedded Deep-Learning include FPGAs \[^19\] but they can have a power consumption of up to 25W which limits their application for battery powered robots.

III. EXPERIMENTAL SETUP

Three platforms are used for evaluating the performance of the CNNs; the Raspberry Pi 3 Model B \[^3\]; the Intel Joule 570X \[^2\] and the Neural Compute Stick \[^1\]. The Neural Compute Stick (NCS) is a low-cost and low-power USB device based on Myriad 2 MA2450 Vision Processing Unit (VPU) \[^9\]. It supports loading networks designed and trained on common deep-learning frameworks like TensorFlow \[^7\] and Caffe \[^14\]. Fig. \[^1\] shows the hardware used for running the CNNs.

TensorFlow and Caffe are used for performing the inference on the Raspberry Pi and the Joule, alternatively, the NCS is attached to each of the platforms over USB for offloading the inference. The power consumption is measured by using the INA219 power monitor IC from Texas Instruments. For the Raspberry Pi and the Joule the INA219 is attached to the power line and then the power is measured before and while performing the inference and the difference is taken as the power required for performing the inferences. For the NCS, the INA219 is attached to the USB connection directly. The inference time is measured by using the system clock.

The version of the used frameworks is shown in Table II. Caffe is compiled with OpenBlas using the branch optimized\_for\_deeplearning. TensorFlow is installed by using the pip package manager. The Raspberry Pi is setup with Raspbian Jessie and the Intel Joule is setup with Ubuntu 16.04. Other frameworks compiled were MXNet, Tiny-dnn, OpenCV and DLib but during the testing they showed unstable behaviour or lack of support for the tested networks so they are not included in the evaluation. The networks used for the NCS were created using the Caffe models as input and setup to use 12 processing cores on Myriad 2 for the inference.

IV. BENCHMARKING

The models and weights for the evaluated networks were downloaded from the TensorFlow and Caffe repositories. The tested architectures are: GoogLeNet \[^23\], AlexNet \[^8\], VGG\_CNN\_F \[^11\], CIFAR10 \[^16\] and Network In Network (NiN) \[^17\]. Table II show details of the tested DNN.

<table>
<thead>
<tr>
<th>Network</th>
<th>Layers</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>GoogLeNet [^23]</td>
<td>27</td>
<td>5 million</td>
</tr>
<tr>
<td>AlexNet [^8]</td>
<td>11</td>
<td>~60 million</td>
</tr>
<tr>
<td>Network in Network [^17]</td>
<td>16</td>
<td>~2 million</td>
</tr>
<tr>
<td>VGG_CNN_F [^11]</td>
<td>13</td>
<td>~500 thousand</td>
</tr>
<tr>
<td>CIFAR10 [^16]</td>
<td>9</td>
<td>~45 thousand</td>
</tr>
</tbody>
</table>

V. OBJECT FOLLOWING

As an application of the CNN inference on low-cost robotics we setup a robot to follow a target autonomously. We use the Raspberry Pi to capture images from the PiCam and then the captured images are sent to the NCS to perform inference. We used the Sinoning Steering robot as it can be purchased online for less than $30USD \[^4\]. Figure \[^4\] shows our setup.

GoogLeNet is used to classify the contents of a template and decide if the shown image corresponds to the desired class as it showed a good accuracy and fast inference. The location of the target is determined by detecting a predefined template. The contents of the template are then send to GoogLeNet to perform inference and determine if the desired target is shown. The target class can be updated on the fly.

Once the desired target class has been recognized, its location and its size within the input image are used to compute the displacement of the robot. The robot is set to try maintain the located object at the centre of the input image and maintain it at a specific size, therefore it moves left/right based on the location of the object and forwards/backwards based on the area of the rectangle.

Table III shows the obtained runtime for each of the stages. The image preprocessing includes the capture of the frame, and image processing for determining the location of the target.
This article presents the first benchmark of CNNs focused on inference time and power consumption for low-cost low-power robotics. We compared the performance of different CNNs using a variety frameworks and embedded platforms. The use of a dedicated hardware accelerator such as the NCS showed how to obtain an inference time up to 4x faster than using the embedded platforms alone while keeping a low-power consumption. As an application of this low-cost low-power environment we setup a robot to follow a target autonomously. By using the NCS we were able process a live stream at 4 FPS which was shown to be enough to allow smooth movement of the robot. Further research includes the evaluation of more CNNs like SSD [18] and YOLO [21] used for object detection. Also as the available frameworks provide support for more types of networks their performances in this kind of low-cost environment need to be tested.

REFERENCES


