Deploying a Caffe Model on OpenMV using CMSIS-NN

Version 1.0
Deploying a Caffe Model on OpenMV using CMSIS-NN

Copyright © 2018 Arm Limited (or its affiliates). All rights reserved.

Release Information

Document History

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Confidentiality</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>15 October 2018</td>
<td>Non-Confidential</td>
<td>First release</td>
</tr>
</tbody>
</table>

Non-Confidential Proprietary Notice

This document is protected by copyright and other related rights and the practice or implementation of the information contained in this document may be protected by one or more patents or pending patent applications. No part of this document may be reproduced in any form by any means without the express prior written permission of Arm. No license, express or implied, by estoppel or otherwise to any intellectual property rights is granted by this document unless specifically stated.

Your access to the information in this document is conditional upon your acceptance that you will not use or permit others to use the information for the purposes of determining whether implementations infringe any third party patents.

This document is provided “AS IS”. ARM PROVIDES NO REPRESENTATIONS AND NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE WITH RESPECT TO THE DOCUMENT. For the avoidance of doubt, Arm makes no representation with respect to, and has undertaken no analysis to identify or understand the scope and content of, patents, copyrights, trade secrets, or other rights.

This document may include technical inaccuracies or typographical errors.

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL ARM BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF ARM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

This document consists solely of commercial items. You shall be responsible for ensuring that any use, duplication or disclosure of this document complies fully with any relevant export laws and regulations to assure that this document or any portion thereof is not exported, directly or indirectly, in violation of such export laws. Use of the word “partner” in reference to Arm’s customers is not intended to create or refer to any partnership relationship with any other company. Arm may make changes to this document at any time and without notice.

If any of the provisions contained in these terms conflict with any of the provisions of any click through or signed written agreement covering this document with Arm, then the click through or signed written agreement prevails over and supersedes the conflicting provisions of these terms. This document may be translated into other languages for convenience, and you agree that if there is any conflict between the English version of this document and any translation, the terms of the English version of the Agreement shall prevail.

The Arm corporate logo and words marked with ® or ™ are registered trademarks or trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. All rights reserved. Other brands and names mentioned in this document may be the trademarks of their respective owners. Please follow Arm’s trademark usage guidelines at http://www.arm.com/company/policies/trademarks.

Copyright © 2018 Arm Limited (or its affiliates). All rights reserved.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

LES-PRE-20349

Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by Arm and the party that Arm delivered this document to.

Unrestricted Access is an Arm internal classification.

Product Status

The information in this document is Final, that is for a developed product.

Web Address

http://www.arm.com
# Contents

1 Overview ......................................................................................................................................................................................... 5  
2 Set up Environment ........................................................................................................................................................................... 6  
3 Define the model .................................................................................................................................................................................. 7  
4 Prepare the dataset .............................................................................................................................................................................. 8  
5 Train the model .................................................................................................................................................................................... 9  
6 Quantize the model ............................................................................................................................................................................ 9  
7 Convert model to binary ..................................................................................................................................................................... 9  
8 Deploy the model on the OpenMV ................................................................................................................................................... 10  
9 Run smile detection ............................................................................................................................................................................. 10  
10 Troubleshooting ............................................................................................................................................................................ 11  
11 Resources ..................................................................................................................................................................................... 12
1 Overview

This guide will walk you through the deployment of a Caffe model to an ultra-low-cost and low-power Arm Cortex-M based processor. For this guide we will be using the OpenMV board, developed by two Arm Innovators, Ibrahim Abdalkader and Kwabena W. Agyeman.

In this guide, you will work through the following steps:

1. Set up Environment
2. Training the Neural Network Model
   • Define the model.
   • Prepare the dataset.
   • Train the model in Caffe.

Note: Training the Neural Network Model is provided for completeness, and is not required to be performed for this guide.

3. Deploy the Model on Arm Cortex-M
   • Quantize the model.
   • Convert model to binary.
   • Deploy on OpenMV.
   • Run smile detection.

At the end of this guide, you will be able to deploy an Arm NN model aimed at recognizing smiling faces on an Arm Cortex-M7 processor.
The following image summarizes the deployment flow that you will go through:

**Deployment flow – Using Caffe**

2 Set up Environment

**Linux machine setup**

Set up your Linux environment by installing a virtual machine as described below. For completeness, we illustrate all the steps to follow to set up your own environment.

**With Virtual Machine**

Note: Installation with Virtual Machine is the recommended option. Installation with Linux will take longer.

1. Install Virtual Box.
   - Install Virtual Box from the USB provided or from this link.
2. Download Image.
   - Download the Virtual Box image using the USB provided or from this link.
3. Import image. Follow this guide.
4. Log into machine.
   - Username: Embedded
   - Password: embedded

or

**Linux**

If required, follow the steps here.
**OpenMV setup**

In the VirtualBox, start the OpenMV IDE from the terminal:

```
./openmvide/bin/openmvide.sh &
```

Note: If not preinstalled, do the following

Install the OpenmMV IDE on the guest OS (the Linux virtual machine). View the full guide [here](#).

Connect board:

1. Connect OpenMV to laptop with USB.
2. To setup the USB connection to the OpenMV board on the virtual machine follow this [USB VM Setup Guide](#).
3. Update firmware if necessary.

Test:

1. Click on the connection icon on the bottom left of the IDE or CTRL+E.
2. Start the program by selecting the play icon on the bottom left of the IDE or CTRL+R.
3. The program should run and display images on the top right without any errors.

Now that the setup is done, we will deploy the chosen model on the Arm Cortex-M7 OpenMV board.

Read the following document for a full OpenMV guide.

**3 Define the model**

Note: You are not asked to do anything in this section. It is only here for completeness.

There are different ways to implement image classification, but recent work in artificial intelligence has shown that a certain type of neural network, called a Convolutional Neural Network (CNN), is particularly good at classifying images. For an introduction to CNNs read this [article](#).

To recognize smiling faces, we will use the Smile model as defined by OpenMV. The network is a three-layer CNN as shown in the figure below:

![CNN diagram](image)

The network is composed of the following layers:

- Convolution layer - responsible for extracting features from the image.
- Dropout layer - responsible for avoiding overfitting by ignoring random nodes during the training phase. For more information read this [dropout article](#).
• Rectified Linear Unit (ReLU) - the activation function responsible for introducing non-linearity in the model. The function returns 0 if it receives any negative input, but for any positive value \( x \) it returns that value back. Read this ReLU article for a more thorough explanation.

• Pooling layer - responsible to progressively reduce the spatial size of the model reducing the number of parameters and the amount of computation in the network, and hence also controlling overfitting. Read more here.

• Inner product (Inp) or fully connected layer.

The above image has been generated using this online tool.

4 Prepare the dataset

The smile dataset that we will use can be found on GitHub.

We will use dataset in
repos/SMILEsmileD/SMILEs/

• negatives
• positives

Check number of images in the two sets:

```
ls repos/SMILEsmileD/SMILEs/negatives/negatives7/ -l | wc -l
ls repos/SMILEsmileD/SMILEs/positives/positives7/ -l | wc -l
```

The dataset consists of ~3000 positive images and ~9000 negative images. To avoid generating a biased model, we would like to have the same number of positive and negative images.

To fix this, we can augment the dataset by using this augmentation script on the positive images. Using this script will increase the number of positive examples by 3x:

Create folder, if it is not already present:

```
mkdir repos/SMILEsmileD/SMILEs/positives/positives_aug/
```

Augment dataset:

```
python2 repos/openmv/tools/augment_images.py --input repos/SMILEsmileD/SMILEs/positives/positives7/ --output repos/SMILEsmileD/SMILEs/positives/positives_aug/ --count 3
```
5 Train the model

Note: Due to time constraints, we won't train the neural network in this guide. This material is only here for completeness.

Currently, support for translating a Caffe model to CMSIS-NN functions is provided. Support for other frameworks (TensorFlow, PyTorch, etc.) will come in the future.

For this guide, the model has been pretrained and can be found here. For a full guide on how to train a model with Caffe, see this tutorial.

The trained model also needs to be converted into an lmdb database. The converted files should be in repos/openmv/ml/cmsisnn/models/smile/smile_database/

6 Quantize the model

Once we have a train mode we need to shrink it to a reasonable size. To do this we use the quantization script from Arm to convert the Caffe model weights and activations from 32-bit floating point to an 8-bit and fixed point format. This will not only reduce the size of the network, but also avoid floating point computations.

The NN quantizer script works by testing the network and figuring out the best format for the dynamic fixed-point representation. The output of this script is a serialized Python (.pkl) file which includes the network's model, quantized weights and activations, and the quantization format of each layer. Running the following command generates the quantized model:

```
python2 repos/ML-examples/cmsisnn-cifar10/nn_quantizer.py --model repos/openmv/ml/cmsisnn/models/smile/smile_train_test.prototxt --weights repos/openmv/ml/cmsisnn/models/smile/smile_iter_*_.caffemodel --save repos/openmv/ml/cmsisnn/models/smile/smile.pk1
```

If you want to learn more about quantization read this blog.

7 Convert model to binary

The final step is to use the OpenMV NN converter script to convert the model into a binary format, runnable by the OpenMV Cam. The converter script outputs a code for each layer type, followed by the layer's dimensions and weights (if any).

On the OpenMV Cam, the firmware reads the binary file and builds the network in memory using a linked list data structure.

Running this command generates the binary model:

```
python2 repos/openmv/ml/cmsisnn/nn_convert.py --model repos/openmv/ml/cmsisnn/models/smile/smile.pk1 --mean /home/embedded/repos/openmv/ml/cmsisnn/models/smile/smile_database/mean.binaryproto --output repos/openmv/ml/cmsisnn/models/smile/smile.network
```
8 Deploy the model on the OpenMV

Download the smile.network directly to the Arm Cortex-M7, on the OpenMV board. To do this use the USB connection.

Note: If the device is not visible on the virtual machine, follow this USB VM Setup Guide.

In this step, you will make use of CMSIS-NN, an open source collection of efficient neural network kernels developed to maximize the performance and minimize the memory footprint of neural networks on Arm Cortex-M processor cores. Neural network inference based on CMSIS-NN kernels achieves 4.6X improvement in runtime/throughput and 4.9X improvement in energy efficiency.

9 Run smile detection

Run the following program on the smile detection Python example on the OpenMV IDE.

Select Files -> Examples -> 25-Machine-Learning -> nn_haar_smile_detection.py

The program enables to identify a person smiling by braking the problem down in two distinct phases:

1. Identify a face in an image using a Haar Cascade classifier.
2. Identify whether the identified face is smiling or not.
3. Display a yellow happy or sad face on the image on the top right.

Optional task: modify the python example

1. Play around with the size of the rectangle used to limit the smile detection area.
2. Add terminal output when a smile is detected.

Optional: final deployable model

Note: This step is not required if the previous steps have been executed successfully.

The final pre-trained and pre-converted models can be found on GitHub here.
10 Troubleshooting

If you get the following errors:

No OpenMV Cams found

This means that the OpenMV is either not connected to the computer or not recognized by the OS:

1. Check that the OpenMV board is connected to the computer via USB.
2. If running on Linux please follow this USB VM Setup Guide.

No module named caffe

```
```
if /home/embedded/caffe/python is not there

>>> exit() export PYTHONPATH=~/caffe/python

Check failed: mdb_status == 0 (2 vs. 0) No such file or directory

Check failed: mdb_status == 0 (2 vs. 0) No such file or directory

Open the file:
repos/openmv/ml/cmsisnn/models/smile/smile_train_test.prototxt

and check that the mean_file and source are pointing to the files in
repos/openmv/ml/cmsisnn/models/smile/smile_database/

OSError: Could not find the file

```python
# Load smile detection network
net = nn.load('/smile.network')

# Load Face Haar Cascade
face_cascade = image.HaarCascade("frontalface")
print(face_cascade)

# FPS clock
clock = time.clock()
while (True):
    clock.tick()
```

Make sure to upload the file to the Arm Cortex-M7 on the OpenMV board.

11 Resources

- Read how Arm is changing machine learning experiences: Project Trillium.
- Why is ML moving to the edge.
- Learn how to train and deploy more models on the OpenMV.
- See CMSIS-NN on Github.
- Explore the OpenMV Tutorial: Low Power Deep Learning on the OpenMV Cam.
- Learn more about the Arm Innovator Program.