Artifact

Masa: Responsive Multi-DNN Inference on the Edge

Cox, Bart; Galjaard, Jeroen; Ghiassi, Amirmasoud; Birke, Robert; Chen, Lydia Y.

DOI
10.1109/PerComWorkshops51409.2021.9431004

Publication date
2021

Document Version
Final published version

Published in
2021 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events, PerCom Workshops 2021

2021 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops) | 978-1-6654-0424-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/PERCOMWORKSHOPS51409.2021.9431004

Artifact: MASA: Responsive Multi-DNN Inference on the Edge

Bart Cox
TU Delft
Delft, Netherlands
b.a.cox@student.tudelft.nl

Jeroen Galjaard
TU Delft
Delft, Netherlands
J.M.Galjaard-1@student.tudelft.nl

Amirmasoud Ghiassi
TU Delft
Delft, Netherlands
s.ghiassi@tudelft.nl

Robert Birke
ABB Research
Baden-Dättwil, Switzerland
robert.birke@ch.abb.com

Lydia Y. Chen
TU Delft
Delft, Netherlands
y.chen-10@tudelft.nl

***Index Terms*—Multiple DNNs inference, mean response time, edge devices, memory-aware scheduling**

I. INTRODUCTION

This artifact is a guideline how the EDGECAFFE framework, presented in [1], can be used. EDGECAFFE is an open-source Deep Neural Network framework for efficient multi-network inference on edge devices. This framework enables the layer by layer execution and fine-grained control during inference of Deep Neural Networks. EDGECAFFE is created to give more fine grained-control over the execution during inference than offered by the original code of Caffe [2]. EDGECAFFE made it possible for MASA to outperform DEEPEYE [3] and normal bulk execution. Besides the core implementation of EDGECAFFE, the repository holds additional tools, *Queue Runner* and *ModelSplitter*, that make more convenient to run experiments and prepare newly trained networks

II. SETUP

In this section we describe different ways to set up the EDGECAFFE framework; using Ubuntu server 18.04.4 (AMD64) or Raspberry Pi (Ubuntu server 18.04.4 ARM64[1]). For this submission, release 1.3.9[2] of EDGECAFFE should be used for evaluation.

The setup for a Raspberry Pi has the following specific instructions: When deploying on a Raspberry Pi, it is important that a swap file exists[3] and swapping is enabled. It is recommended to use a swap file of 8 GB.

1) Swapping can be enabled in the grub file by adding the `cgroup_enable=memory` argument.
2) By default swapping is turned off for Raspberry Pi's. This can be enabled by running the shell command `sudo swapon -a`

To set up EDGECAFFE:

1) Clone the repository[2]
2) In the repository folder, run the shell script `bash setup.sh`[4]
3) To set up the prepared networks, run the shell script `bash install_models.sh`[5].
4) To compile and install run the shell script `bash compile.sh`
5) Set the correct owner to the installed owner by executing the following command in the terminal: `sudo chown -R $USER:$USER /opt/edgecaffe`

III. USAGE

After EDGECAFFE has been installed successfully, the binaries can be accessed from the folder `/opt/edgecaffe`. The experiments can be found in the folder *experiments*. Yaml files describe the structure

[1]http://old-releases.ubuntu.com/releases/18.04.4/ubuntu-18.04.4-preinstalled-server-armhf+raspi4.img.xz

[2]https://github.com/bacox/edgecaffe/tree/v1.3.9

[3]https://linuxize.com/post/how-to-add-swap-space-on-ubuntu-18-04/

[4]Execute `chmod +x <filename>` if the script is not executed with bash

[5]The models used in this script are located at https://gitlab.com/bacox/edgecaffe-models/-/tree/v1.1

and parameters of the experiments. To automate the execution of all the experiments, the script *Queue Runner* is included. With *Queue Runner* we can queue a set of experiments that will be executed one after the other.

A. Basic Example

To set up a basic example with *Queue Runner*, the next commands should be executed in the `/opt/edgecaffe` folder:

1) To generate the experiment files run `bash scripts/percom/gen_basic_queue_file.sh`
2) Add the generated file to the Queue Runner by executing: `python3 scripts/percom/queue_runner.py add --file=example.tmp.txt`
3) Start the Queue Runner to process all the experiments in the background: `python3 scripts/percom/queue_runner.py run`

B. PerCom experiments

In order to set up and run the experiments of the Masa paper with *Queue Runner*, the next commands should be executed in the `/opt/edgecaffe` folder:

1) To generate the experiment files run `bash scripts/percom/gen_percom_queue_file.sh`
2) Add the generated file to the Queue Runner by executing: `python3 scripts/percom/queue_runner.py add --file=percom-rpi-4.tmp.txt`
3) Start the Queue Runner to process all the experiments in the background: `python3 scripts/percom/queue_runner.py run`

C. Results

The outcomes of the experiments are written to the folder `/opt/analysis`. Each experiments results in the following files:

- Arrival file: All the networks that have arrived during this experiment.
- Network statistics file: The arrival, waiting, and execution time of each arrived network.
- Layers file: The arrival, waiting, and execution time of each layer in each arrived network.
- Worker file: For each worker and individual file is saved with the duration the worker is busy and the duration the worker is idle.

IV. New networks

The EdgeCaffe framework offers 10 networks that are prepared to use. New networks can be altered in order to be executed with EdgeCaffe. A network folder contains the following elements:

- Prototxt file describing the structure of the network.
- Caffemodel file containing the trained weights and parameters of the network.
- `description.yaml` describing the elements of a network in more detail.
- Partials folder containing the caffemodel files for each of the layers separately.

A network trained with Caffe already has a prototxt and a caffemodel file. Two additional steps needs to be taken to prepare a new network for execution with EdgeCaffe. First, information describing the network is put in the file `description.yaml`. The tool *ExtendNetworkDescription* (included in EdgeCaffe) can used to make this process easier. Secondly, the caffemodel file needs to be split into separate caffemodel files for each layer in the network. The tool *ModelSplitter* can be used to automate this process.

When a trained network is prepared successfully, it can be used by placing it in the network folder and refer to the model in the configuration file of the experiment.

V. Acknowledgements

This work has been partly funded by the Swiss National Science Foundation NRP75 project 407540_167266.

References

[1] B. Cox, J. Galjaard, A. Ghiassi, L. Y. Chen, and R. Birke, "Masa: Responsive multi-dnn inference on the edge," in *IEEE PerCom*, p. to appear, 2021.

[2] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *ACM International Conference on Multimedia*, pp. 675–678, 2014.

[3] A. Mathurz, N. D. Lanezy, S. Bhattacharyaz, A. Boranz, C. Forlivesiz, and F. Kawsarz, "DeepEye: Resource efficient local execution of multiple deep vision models using wearable commodity hardware," *MobiSys*, pp. 68–81, 2017.