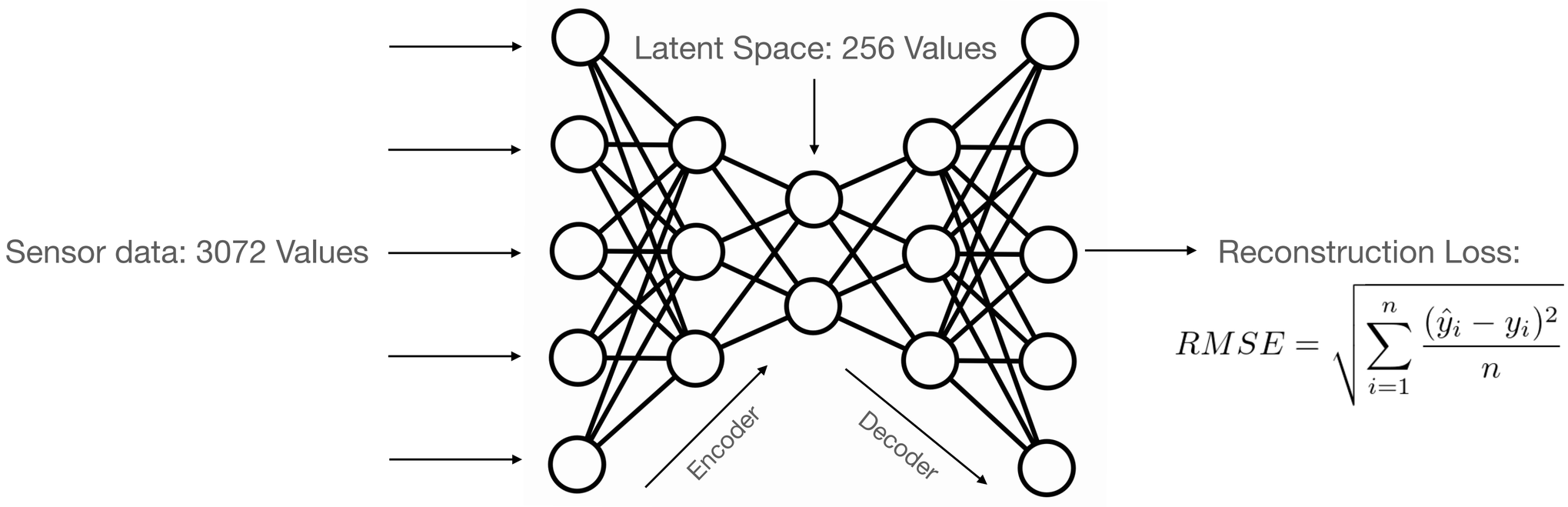
Hardware acceleration of Neural Networks on Edge Devices

Inference performance on the Google Edge TPU and Intel's Neural Compute Stick 2

Jannis Wolf, B.Sc., 22.7.2021

Anomaly Detection

Using an under complete autoencoder



Undercomplete Autoencoder[1] for anomaly detection [2]

- Reconstruction loss over threshold —> Anomaly detected

Anomaly Detection using Autoencoders

Model setup

- Input: Sensor data (3072,)
- Encoder: Fully Connected Layer
  - Weights: (3072, 256)
- Decoder: Fully Connected Layer
  - Weights: (256, 3072)
- Activation function: Leaky ReLU
- Essentially two big matrix multiplications

Input tensor

$$\begin{pmatrix} w_{1,1} & \cdots & w_{n,1} & w_{n+1,1} \\ \vdots & \ddots & \vdots & \vdots \\ w_{1,m} & \cdots & w_{n,m} & w_{n+1,m} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_n \\ 1 \end{pmatrix} = \begin{pmatrix} \hat{y}_1 \\ \vdots \\ \hat{y}_m \end{pmatrix}$$

Weight tensor Activation

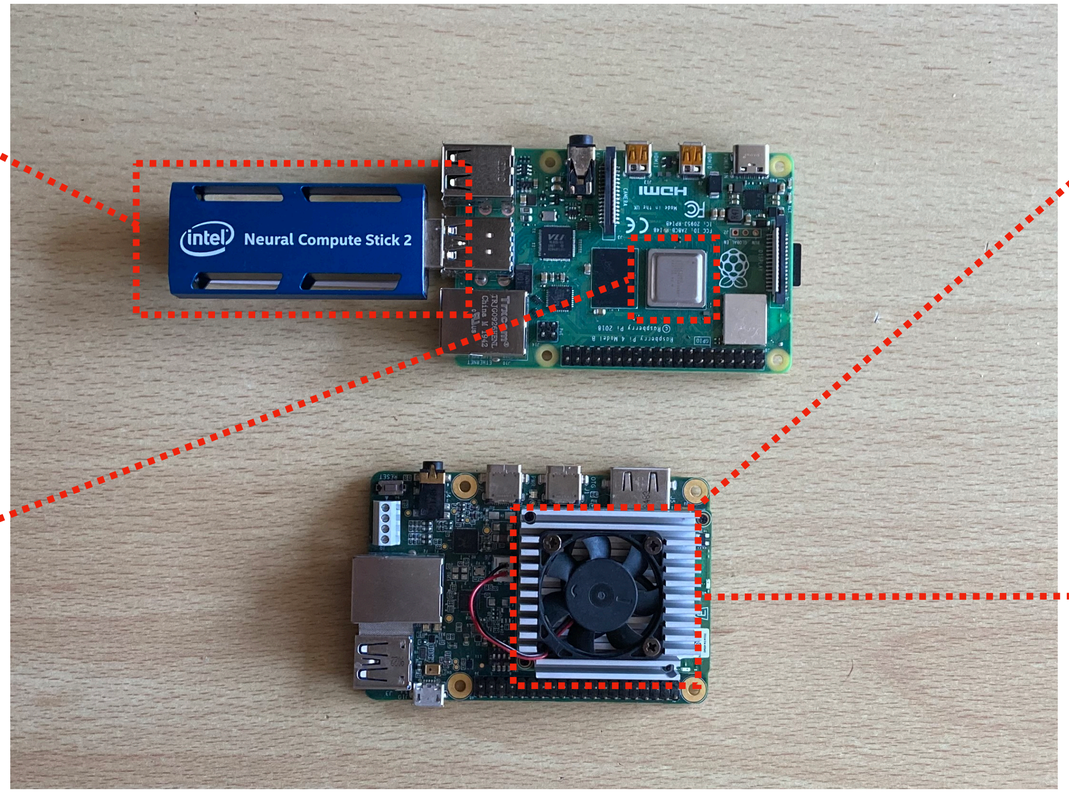
The Competitors

Intel Neural Compute Stick 2

- Openvino
- Float16

RPi 4, Cortex-A72

- Native Tensorflow
- Float32



Up: RPi 4 with Intel NCS2, Down: Google Dev Board

Google Coral Edge TPU

- Tensorflow Lite
- Int8

Google Coral Cortex-A53 CPU

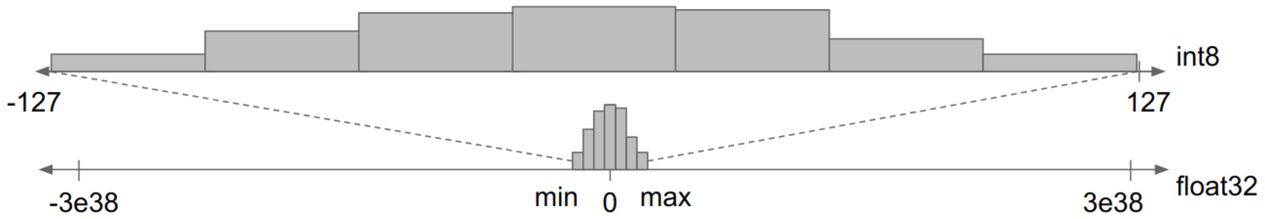
- Tensorflow Lite
- Float16 and Int8

General Measurement Setup

- Inference on several devices measured
  - Inference time per sample in ms
  - Reconstruction Loss (RMSE)
- Testset contains 100 samples
- Mean over test set -> stable training, error bars were omitted
- Loading the model is excluded, loading tensor to model is included

A word on precision

- Baseline neural networks: float32
- Intel NCS2 Myriad VPUs: float16
- Google EdgeTPU: int8



Float32 to Int8 Quantisation [3]

- Resulting reconstruction loss

	Float32	**Float16**	**Int8**
RMSE	0,00025	0,00026	1,375

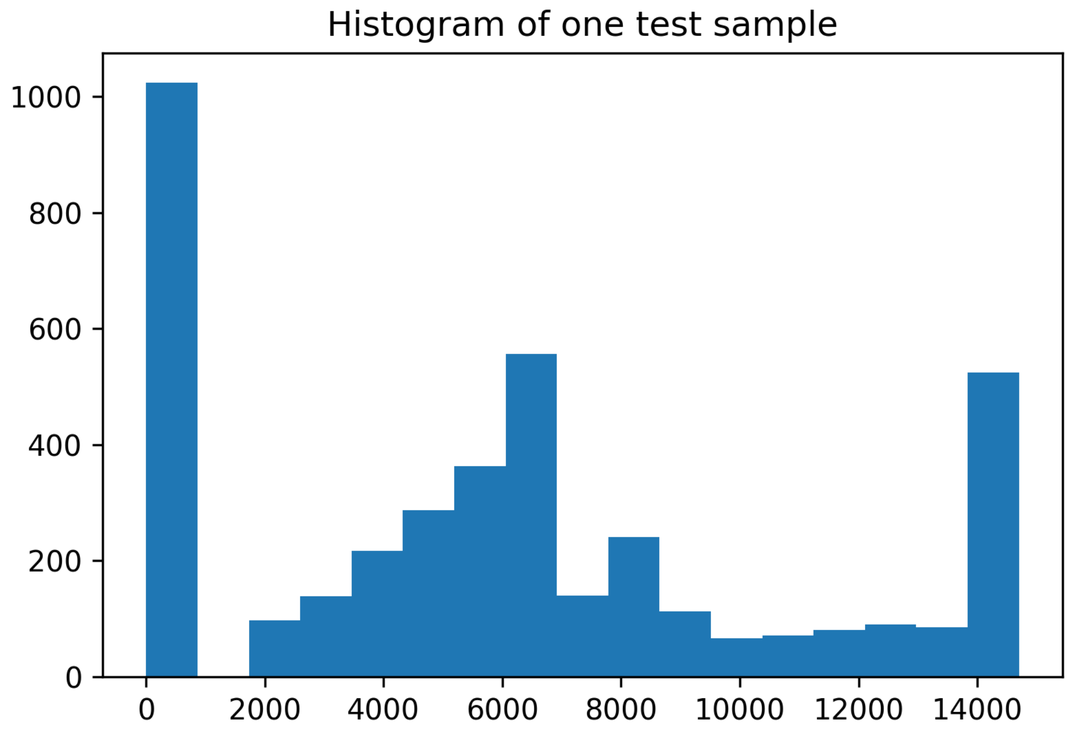
Explanation for high error

1. Range of values to big for such a radical quantisation

—> Retraining after quantisation

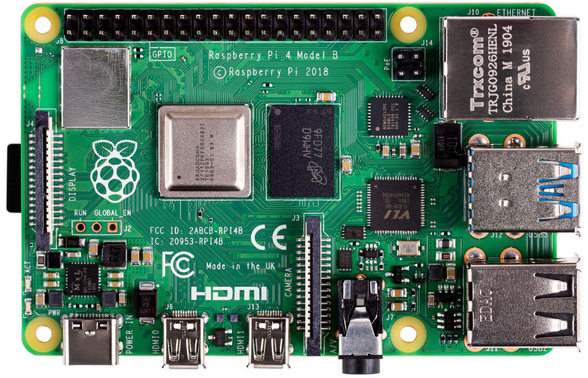
2. Autoencoding is a regression task

—> Quantisation for classification of pictures is easier as pixel values often range only in uint8



Intel Neural Compute Stick 2

Intel NCS 2 [4]



Raspberry Pi 4b [5]

Intel Neural Compute Stick 2

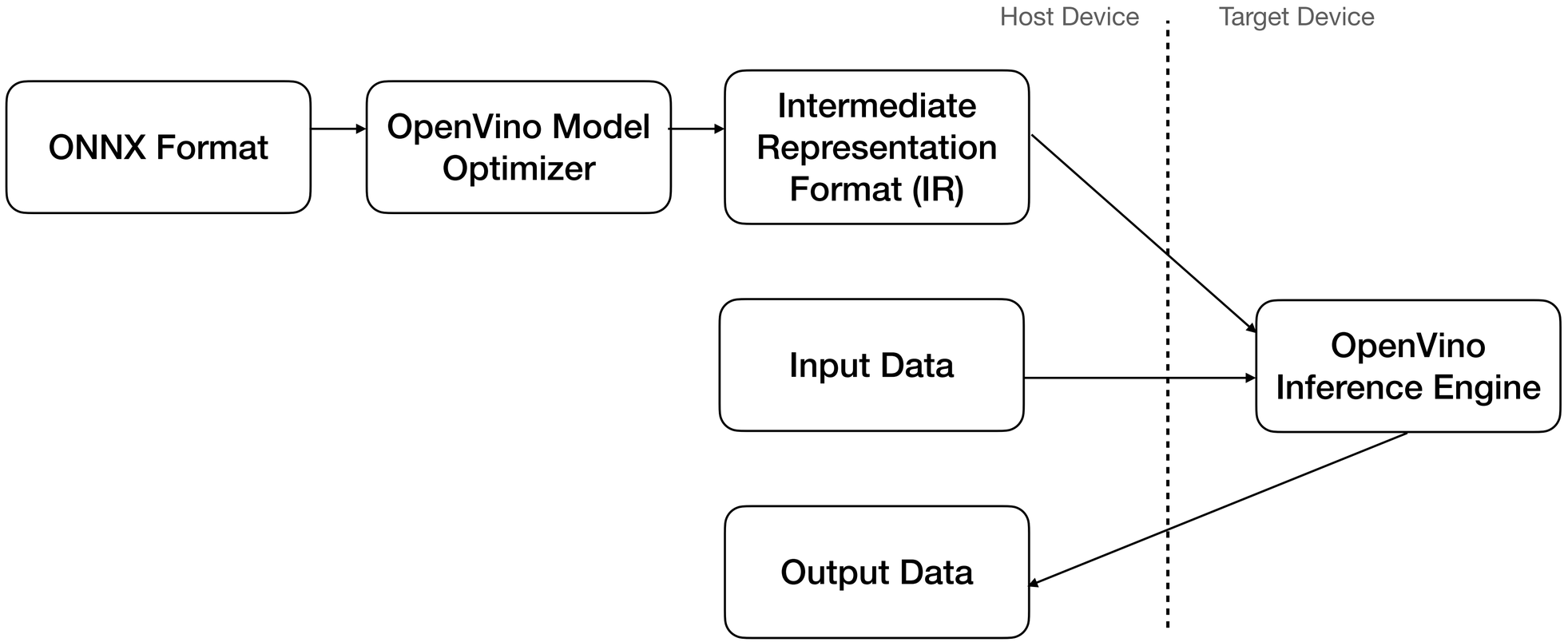
- Inference on MYRIAD VPU
- ONNX model
- OpenVino inference engine

Raspberry Pi 4

- Inference on Cortex A72 CPU
- Native Tensorflow 2.3

Preparing NCS2 Model

Pipeline to model inference in OpenVino



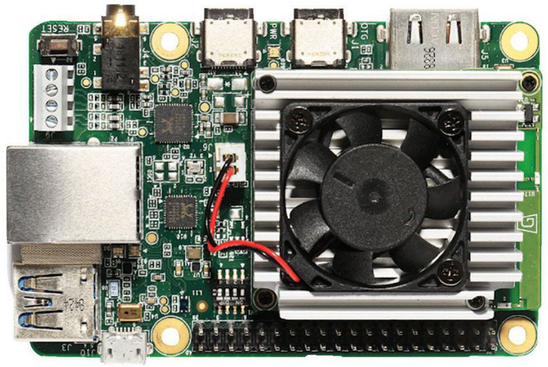
Performance Intel NCS2 vs RPi4

- Average Inference time for one sample in ms

RPi 4	Intel NCS2
104,26 ms	2,16 ms

- Speed Up of 48.2 (while no significant accuracy loss)
- OpenVino is extremely fast
- Powerful and cheap upgrade of inference feature for edge devices

Google Coral



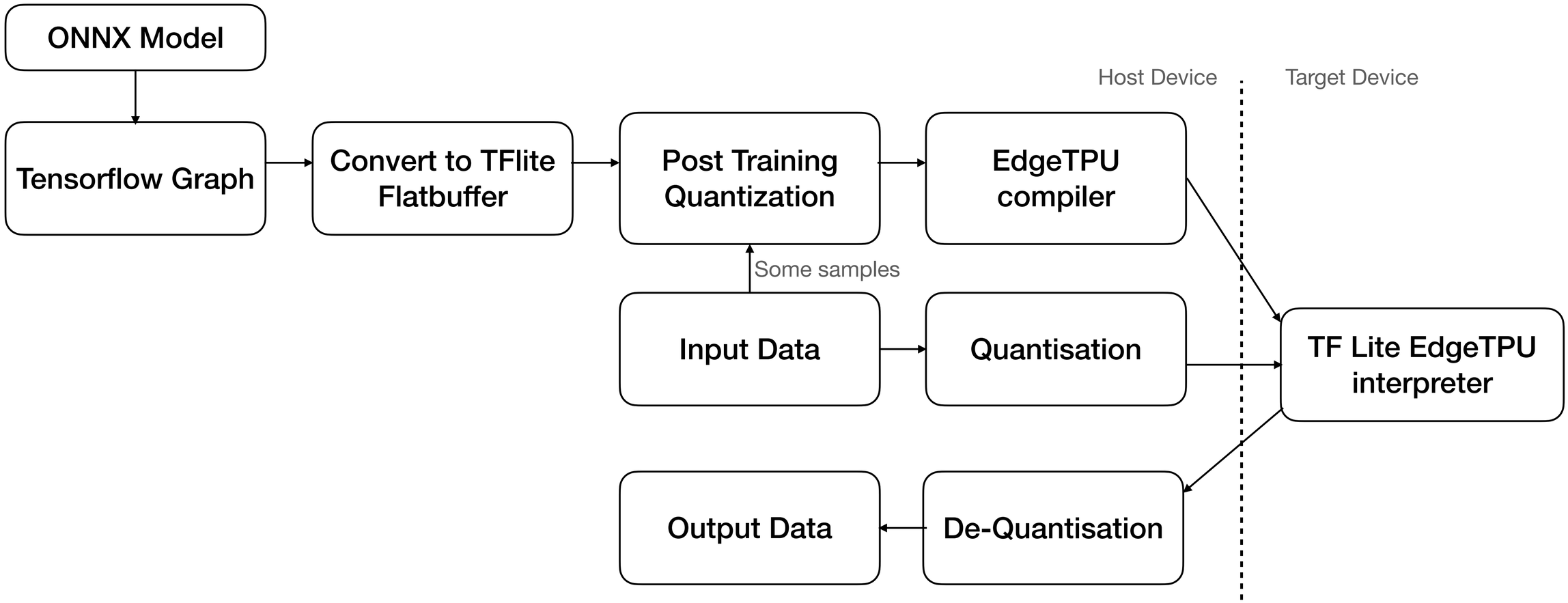
Google Coral Dev Board [6]

Google Edge TPU

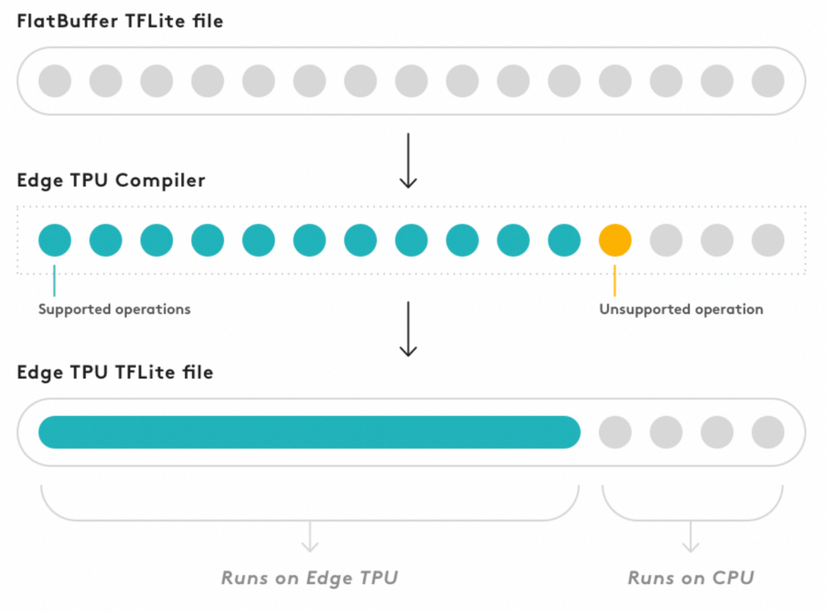
- Google’s EdgeTPU
- Quad Cortex-A53 CPU
- Tensorflow Lite with Int8 quantisation on TPU and CPU
- Reference no quantisation inference on CPU

Preparing EdgeTPU Model

Pipeline to model inference on a Google EdgeTPU



EdgeTPU Compiler



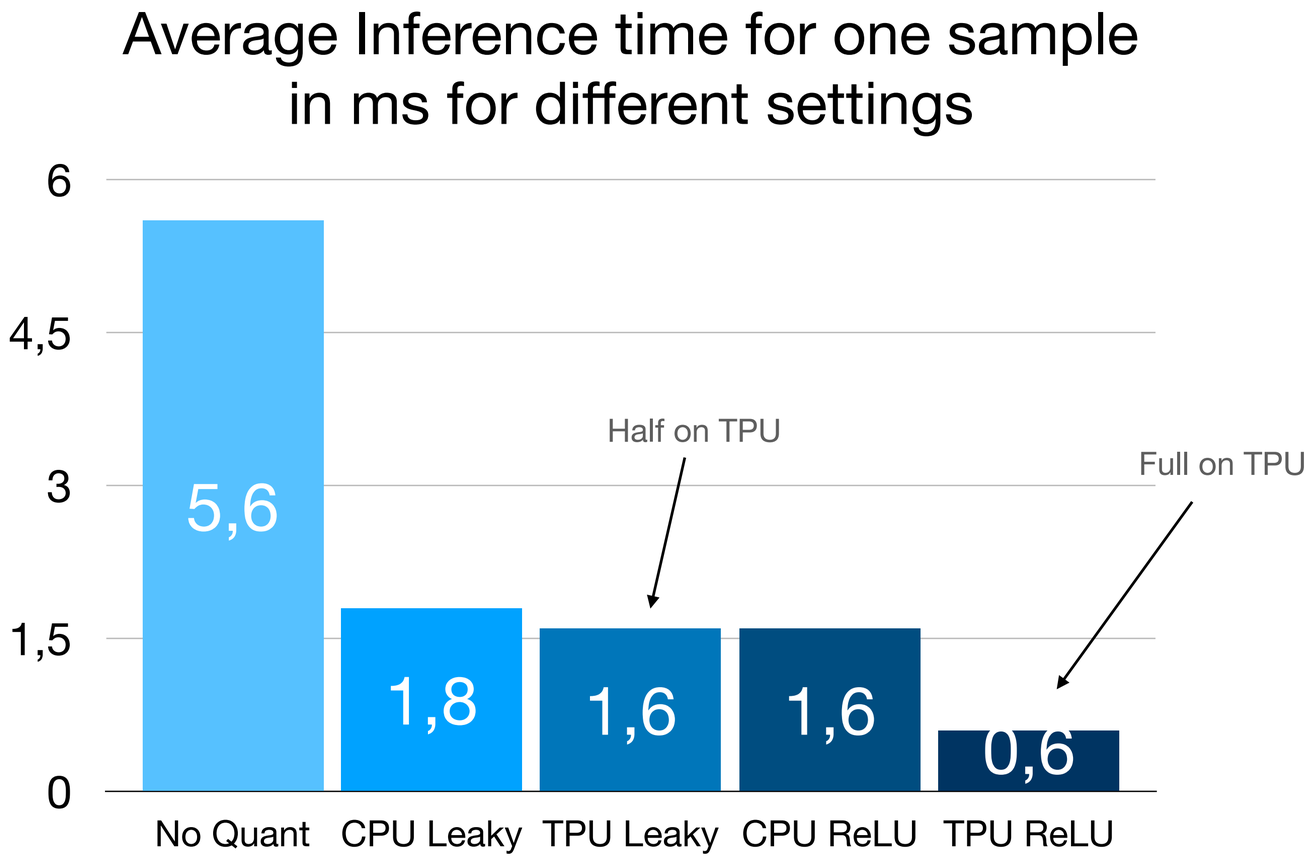
Conversion from TF lite model to Edge TPU lite model [7]

Converting the autoencoder

- Leaky ReLU unsupported
- Second matrix multiplication on CPU
- Changing activations to supported ReLU function

—> Second model with ReLU activation runs fully on EdgeTPU

Performance Google Coral TPU vs CPU



- TF Lite much faster than native Tensorflow
- TPU Leaky: Data transfer decreases performance
- TPU ReLU: Speed Up of 2.6 to CPU
- Speed Up of 9.3 to no quant model

—> Trade-off between performance and accuracy

Conclusion

- Intel NCS2 highly outperforms ARM chips on low power devices
  - Cheap inference feature upgrade
  - OpenVino ensures compatibility of code with NCS2, CPUs and GPUs
  - Tested for deployment in space for onboard processing [8]
- Google Coral showed performance gains for this task
  - Benefits show more on bigger convolutional networks like MobileNet [9]
  - Promising results on vision tasks as mobile chip
  - Easy integration into existing mobile tasks through TF Lite

Thanks for your attention!

Any questions?

Jannis Wolf, B.Sc., 22.7.2021
Github: https://github.com/JannisWolf/evaluating-edge-accelerators

Sources

[1] Charte et al., A practical tutorial on autoencoders for nonlinear feature fusion: Taxonomy, models, software and guidelines, 2018

[2] Anomaly Detection Using Autoencoders with Nonlinear Dimensionality Reduction, Sakurada et al., 2014

[3] https://blog.tensorflow.org/2020/04/quantization-aware-training-with-tensorflow-model-optimization-toolkit.html (accessed 22.7.21)

[4] https://www.intel.com/content/dam/support/us/en/documents/boardsandkits/neural-compute-sticks/NCS2_Datasheet-English.pdf (accessed 19.5.21)

[5] https://de.wikipedia.org/wiki/Raspberry_Pi (accessed 22.7.21)

[6] https://coral.ai/products/dev-board/ (accessed 22.7.21)

[edge] https://coral.ai/docs/edgetpu/models-intro/ (accessed 22.7.21)

[8] https://www.techtimes.com/articles/253524/20201021/intel-movidius-myriad-2-vpu-takes-first-trip-space-aboard.htm (accessed 22.7.21)

[9] https://coral.ai/docs/accelerator/get-started/#3-run-a-model-on-the-edge-tpu (accessed 22.7.21)