Hands-on example: Performance analysis with Streamline

Before profiling the application with Arm Streamline, it is worth having a closer look at the code in the mnist-demo application. Open either one of the mnist_tf_convol.cpp or mnist_tf_simple.cpp.

Following the comments in the code should suffice to describing the main phases.

1. Load and parse the MNIST data
The helper function in mnist_loader.hpp scans the file in the dataset and returns a *MnistImage* struct with two fields: the label and an array of pixel values.

2. Import the TensorFlow graph
You can import a TensorFlow graph from both text and binary Protobuf formats.
Importing a graph consists of binding the input and output points of the model graph.
You can find these points by visualising the model in TensorBoard.
Note: After this step, the code is common regardless of the framework that you started with.

3. Optimise for a specific compute device
Arm NN supports optimisation for both CPU and GPU devices.
It is easy to specify the device when creating the execution runtime context in the code.

4. Run the graph
Running the inference on the chosen compute device is performed through the *EnqueueWorkload()* function of the context object.
The result of the inference can be read directly from the output array and compared to the MnistImage label we read from the data file.

Compile the Convolutional Neural Network graph that we will profile in Streamline.

```
$ cd mnist-demo
$ make
g++ -O3 -std=c++17 mnist_tf_simple.cpp -o mnist_tf_simple \
      -I/home/arm01/armnn-devenv/armnn/include -I/home/arm01/armnn-
devenv/pkg/boost/install/include \
      -L/home/arm01/armnn-devenv/armnn/build -larmnn -larmnnTfParser
-lpthread
g++ -O3 -g -std=c++17 mnist_tf_convol.cpp -o mnist_tf_convol \
      -I/home/arm01/armnn-devenv/armnn/include -I/home/arm01/armnn-
devenv/pkg/boost/install/include \
      -L/home/arm01/armnn-devenv/armnn/build -larmnn -larmnnTfParser
-lpthread
Make sure to add Arm NN: export LD_LIBRARY_PATH=/home/arm01/armnn-
devenv/armnn/build
```

This is a good time to compare mnist_tf_convol and mnist_tf_simple and see what is different.

Run MNIST inferences

```
# Optimisation modes: 0 for CpuRef, 1 for CpuAcc, 2 for GpuAcc
# Input size: 1 to 2000 (number of images to predict)
$ ./mnist_tf_convol 1 10
```

Try diffent program configurations and compare the execution times

```
# 10 images on unoptimised CPU device
$ time ./mnist_tf_convol 0 10

# 10 images on optimised CPU device
$ time ./mnist_tf_convol 1 10

# 100 images on GPU device
$ time ./mnist_tf_convol 2 100
```

Use Streamline to connect and profile the application

Streamline is a performance analyzer for software running on Arm processors.
Note: Instructions following assume that *Streamline is available* on the user machine.

Step 0. Ensure the application we want to profiling is running correctly

The Arm NN library is a **.so** file located by *LD_LIBRARY_PATH* environment variable in ~/.bashrc.
The application binary is compiled as instructed in previous steps.
You should see output similar to this:

```
$ ./mnist_tf_convol 1 10
```

```
Optimisation mode: CpuAcc
#1 | Predicted: 7 Actual: 7
…
#10 | Predicted: 9 Actual: 9
Prediction accuracy: 100%
```

Step 1. Start Streamline gatord on the board

The gator daemon is available under $HOME/armnn-devenv/gator/daemon/gatord
Adjust the path as needed for your system.

Run using **sudo** to receive access to all hardware counters and use -a to allow gator to issue commands.

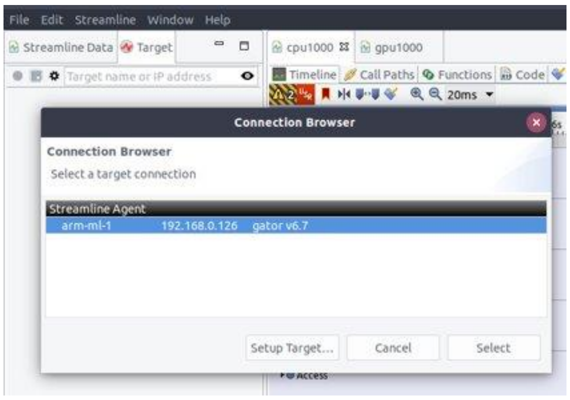
```
$ sudo $HOME/pkg/gator/daemon/gatord -a
```

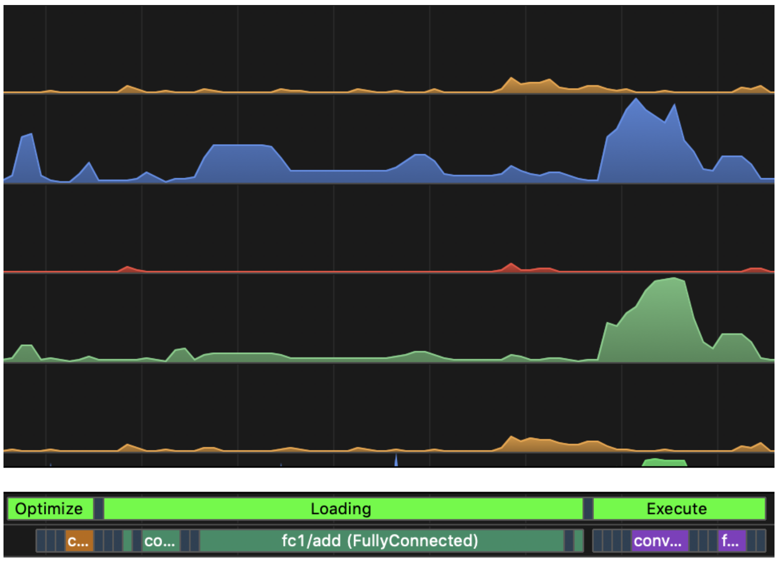
Step 2. Connect Streamline to the target board

Step 3. Start a capture in Streamline -> Run the application -> Stop the capture in Streamline

Run the application manually from the command line for 1000 inferences after the capture is started and end the capture when the program is done.

```
$ ./mnist_tf_convol 1 10000
```





Streamline shows a detailed, annotated timeline view.

Some observations:

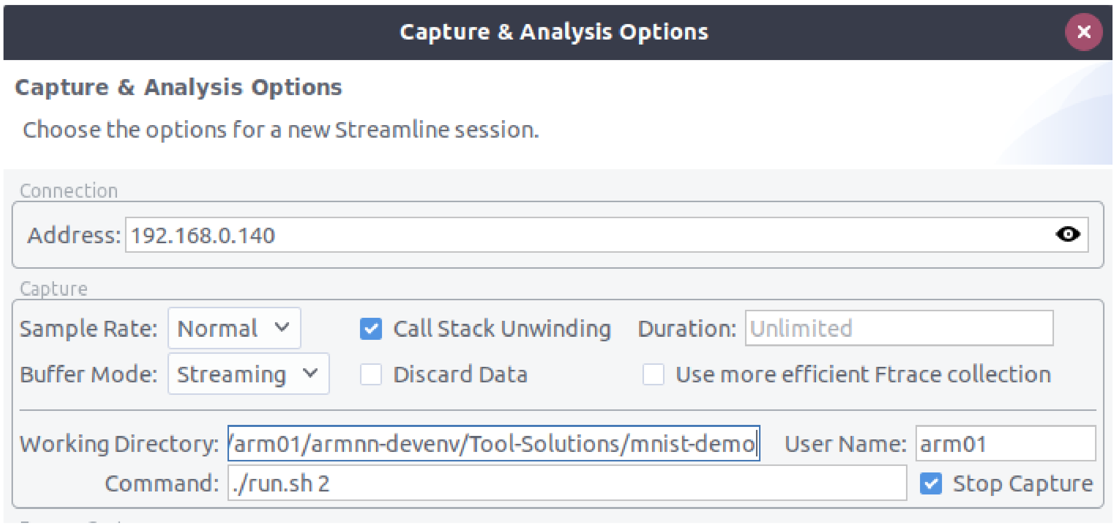
- CPU workload
- Correlation between the phases and the functions called
- Correlation between the functions and the kernels

Automate the launch and capture

To automate the launch and capture use the Streamline Capture & Analysis Options dialog to specify a working directory and a Command to run.

There is a run.sh script in the mnist-demo/ directory which can be the command to launch.

Also, click the “Stop Capture” checkbox to stop the capture when the run is over.



Capture & Analysis Options

Capture & Analysis Options

Choose the options for a new Streamline session.

Connection

Address: 192.168.0.140

Capture

Sample Rate: Normal [✓ Call Stack Unwinding] Duration: Unlimited

Buffer Mode: Streaming ☐ Discard Data ☐ Use more efficient Ftrace collection

Working Directory: /arm01/armnn-devenv/Tool-Solutions/mnist-demo User Name: arm01

Command: ./run.sh 2 [✓ Stop Capture]

Study the run.sh script and experiment with different options and see different results for the CPU and GPU profile.