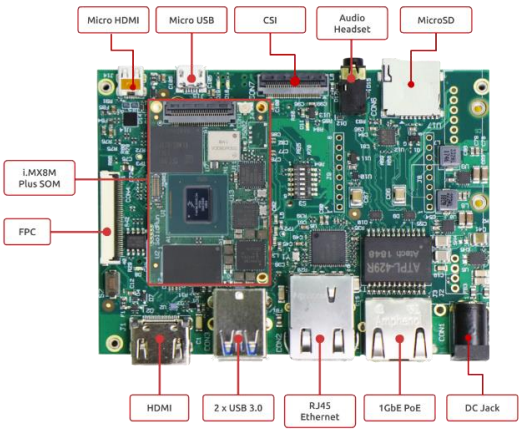
Getting a HummingBoard up and running for federated learning

This is a guide to running federated learning on a HummingBoard Pulse. The board is manufactured by SolidRun and has SOM model NXP i.MX8MP. I document the steps I took that worked for me, some that didn't and highlight some issues. I have essentially followed advice in the quick start guide with some issues arising on the way that I document below. More information may be found in the community forum of SolidRun.

Equipment used:

- HummingBoard Pulse card
- Power adapter
- Ethernet cable
- Micro USB to USB cable
- Micro SD card (16 GB)
- Linux PC

Below you can see the initial setup. The cable to the lower right is the micro USB to USB connected to a Linux PC for serial connection.



Writing the Linux image to the SD card

Insert the Micro SD card in your Linux PC. You can for instance use a Micro SD card reader with USB connection. Download the image as follows:

```
wget https://solid-run-images.sos-de-fra-1.exo.io/IMX8/Debian/sr-imx8-debian-bullseye-20220407-cli-imx8mp-sdhc.img.xz
```

Other releases may be found here. I am using the Debian Release for i.MX8, there are alternatives to this such as the Yocto project. The next step is to flash the image to the SD card. I followed the steps below. Some of the commands may require a sudo in front.

- Extract image with
  `xz -d sr-imx8-debian-bullseye-20220407-cli-imx8mp-sdhc.img.xz`
- Check where the SD card is mounted with fdisk -l. You will see a similar size device mounted at /dev/sdb or similar. There may be partitions at /dev/sdb1 and so on. My SD card was mounted as /dev/sdb and a partition at /dev/sdb1.
- Unmount partition with umount /dev/sdb1.

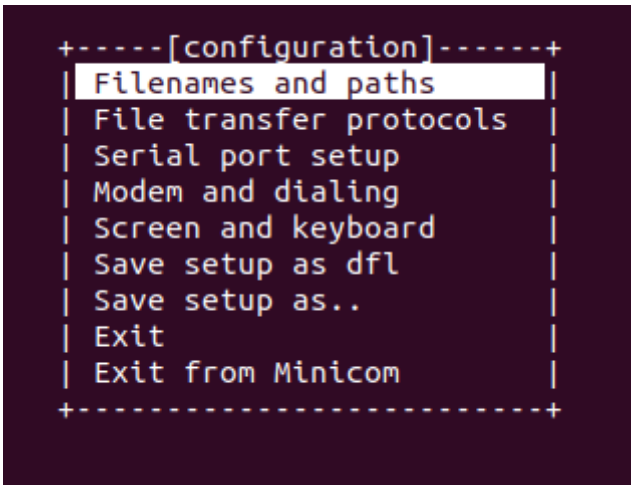
- Be sure to get the right disk in the command below (/dev/sdb in my case). The operation may erase data on your other drives if you get it wrong.
- Write image to SD card with `dd bs=4k conv=fsync if=sr-imx8-debian-bullseye-20220407-cli-imx8mp-sdhc.img of=/dev/sdb`

Resizing the partition and or adding a swap partition to the SD card (which may be done with gparted) had a tendency to break the image so I skipped this. If you have issues with ending up with a too small partition and have problems with resizing, contact me and we can try to work it out.

Setting up serial connection

As far as I know, the Debian image above does not come with a desktop environment and so we can either connect a keyboard and monitor to the card and use the Linux terminal or we can connect to the card using serial connection and another PC. Here is how I did the latter.

- Connect the Micro USB cable to the USB port of the Linux PC. Run dmesg to find out the USB we are connected to. One of the output lines will read something like `[518569.331514] usb 1-6: FTDI USB Serial Device converter now attached to ttyUSB0`
- Install minicom and start it with minicom -s. In the menu, select Serial port setup.



- In A – Serial Device: fill in the USB from above (/dev/ttyUSB0 in my case).

- You can save settings and make them default by choosing Save setup as dfl.
- Connect power cable to card and watch the system boot.
- Login with user “debian” and password “debian”.

Resetting the hardware clock

When firing up the HummingBoard and trying to install some additional software I got an issue that certificates for the Debian package sources were “not yet valid”. I could see this for example by doing sudo apt-get update. It turned out the hardware clock and system clock were several years behind. You can check the system clock with date and hardware clock with sudo hwclock –show. You can set the system time manually using timedatectl set-time YYYY-MM-DD HH:MM:SS. To copy system time to hardware time you can do hwclock –systohc.

Installing some software

At this stage I used an ethernet cable connected to a router for internet access. This worked right out of the box. Wifi connection is discussed below.

The Debian installation comes shipped with Python 3.9.2. I managed to install some software (without any problems) as follows:

- sudo apt-get install python3-pip
- sudo apt-get install emacs
- sudo pip install flwr

When trying to install TensorFlow I ran into trouble though, more on that below.

Transferring files

To transfer files such as scripts or training data from your Linux PC, you can use the serial connection and minicom as described above.

- Install lrzsz with sudo apt-get install lrzsz
- In your minicom session, go to your desired file destination and do rz -bZ
- In your minicom session do CTRL-A Z
- Choose S for Send files and zmodem in the [Upload] menu. Now choose file on your Linux PC.

Tensorflow and federated learning

Previously we have used TensorFlow together with the Flower framework for federated learning. When attempting to install TensorFlow with pip on the HummingBoard, pip issued an oom-kill (out of memory kill) during wheel download. The specs say that memory on the card is “up to 4 GB” and doing grep MemTotal /proc/meminfo on the HummingBoard gives MemTotal: 1004056 kB, roughly 1 GB. I am not sure what the issue is during the TensorFlow install since pip kills the process after only a few hundred MB:s have been downloaded but either way I suspect it has to do with the fact that I

have no swap partition on my SD card. I managed to install TensorFlow by designating a (temporary) swap file on my disk as follows, rather than having a swap partition:

- `sudo fallocate -l 2G /swapfile`
- `sudo chmod 600 /swapfile`
- `sudo mkswap /swapfile`
- `sudo swapon /swapfile`

To check that everything looks ok you can do:

- `sudo swapon –show`

I choose 2GB but a 1 Gb file should be enough. After taking the above steps, installing TensoFlow with pip install tensorflow worked without issues.

Alternatives to Flower and TensorFlow

While having issues with the size of TensorFlow I looked briefly at alternatives:

- TensorFlow Lite with Flower.
- TensorFlow Federated.

As far as I can see, TensorFlow Federated requires TensorFlow and is not made to work together with TensorFlow Lite alone. This setup is therefore similar to using Flower with TensorFlow.

TensorFlow Lite is much smaller than TensorFlow and also comes installed in the NXP eIQ Toolkit. However, it is not (yet) fully integrated with Flower and it would take some work to get it up and running. In fact, TensorFlow Lite is built primarily for inference, that is using models rather than training them. There is however functionality in TensorFlow Lite for fine tuning pretrained models (model personalization). This means that additional training of a pretrained model is done on the edge device, and this can be used to do federated learning. I view this as a kind of hack though and in my opinion it is better to use TensorFlow or wait for updates to Flower.

Federated Learning with Flower on the HummingBoard

I managed to perform federated learning with the HummingBoard as participant. The first experiment involves two clients; one client is running on the HummingBoard and the other client is running on my laptop together with the central server. Also, at this stage, I use the ethernet connection on the board and data collection and feature extraction are prepared beforehand. At a later stage we will test the camera for the HummingBoard as well as the wireless connection.

The minimal server script server.py looks as follows:

```
import flwr as fl
# Start Flower server
fl.server.start_server(
    server_address="0.0.0.0:8080",
    config=fl.server.ServerConfig(num_rounds=100))
```

Note the choice of port 8080.

For the client running on same laptop as the server we have a client script client.py as follows:

```
import os
import flwr as fl
```

```
import tensorflow as tf
import pickle
import sys
# Make TensorFlow log less verbose
os.environ["TF_CPP_MIN_LOG_LEVEL"] = "3"
args = sys.argv
if len(args) == 1:
    p = 5.0
else:
    p = float(args[1])

model = tf.keras.Sequential()
model.add(tf.keras.layers.Dense(10, activation='relu'))
model.add(tf.keras.layers.Dense(10, activation='relu'))
# model.add(tf.keras.layers.Dropout(0.5))
model.add(tf.keras.layers.Dense(1, activation='sigmoid'))
model.compile("adam", "mean_squared_error", metrics=["accuracy"])
model.build([1,2])
local_data = pickle.load(open("local_data.p", 'rb'))
X_train, y_train, X_test, y_test = local_data[p]

# Define Flower client
class CifarClient(fl.client.NumPyClient):
    def get_parameters(self, config):
        return model.get_weights()

    def fit(self, parameters, config):
        model.set_weights(parameters)
        model.fit(X_train, y_train, epochs=1) #,
batch_size=len(X_train))
        return model.get_weights(), len(X_train), {}

def evaluate(self, parameters, config):
    model.set_weights(parameters)
    loss, accuracy = model.evaluate(X_test, y_test) #,
batch_size=len(X_test))
    return loss, len(X_test), {"accuracy": accuracy}

# Start Flower client
fl.client.start_numpy_client(server_address="127.0.0.1:8080",
client=CifarClient())
```

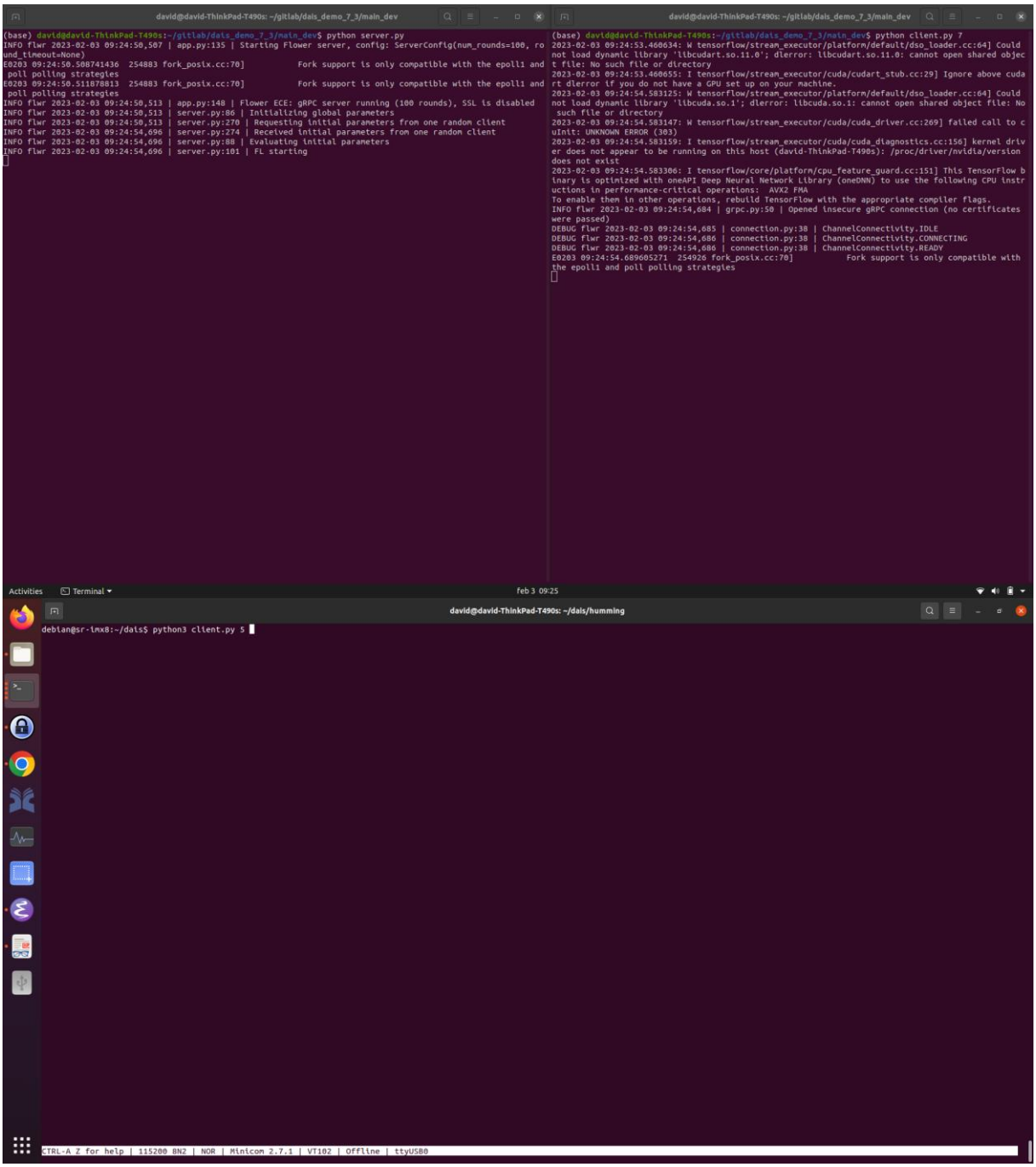
Note the choice of server IP address 127.0.0.1 and port 8080. The client on the HummingBoard can be exactly the same except for the server address. To find the IP address of your machine you may do hostname -I and put server_address=<your-ip-address>:8080.

To initiate the learning simply do:

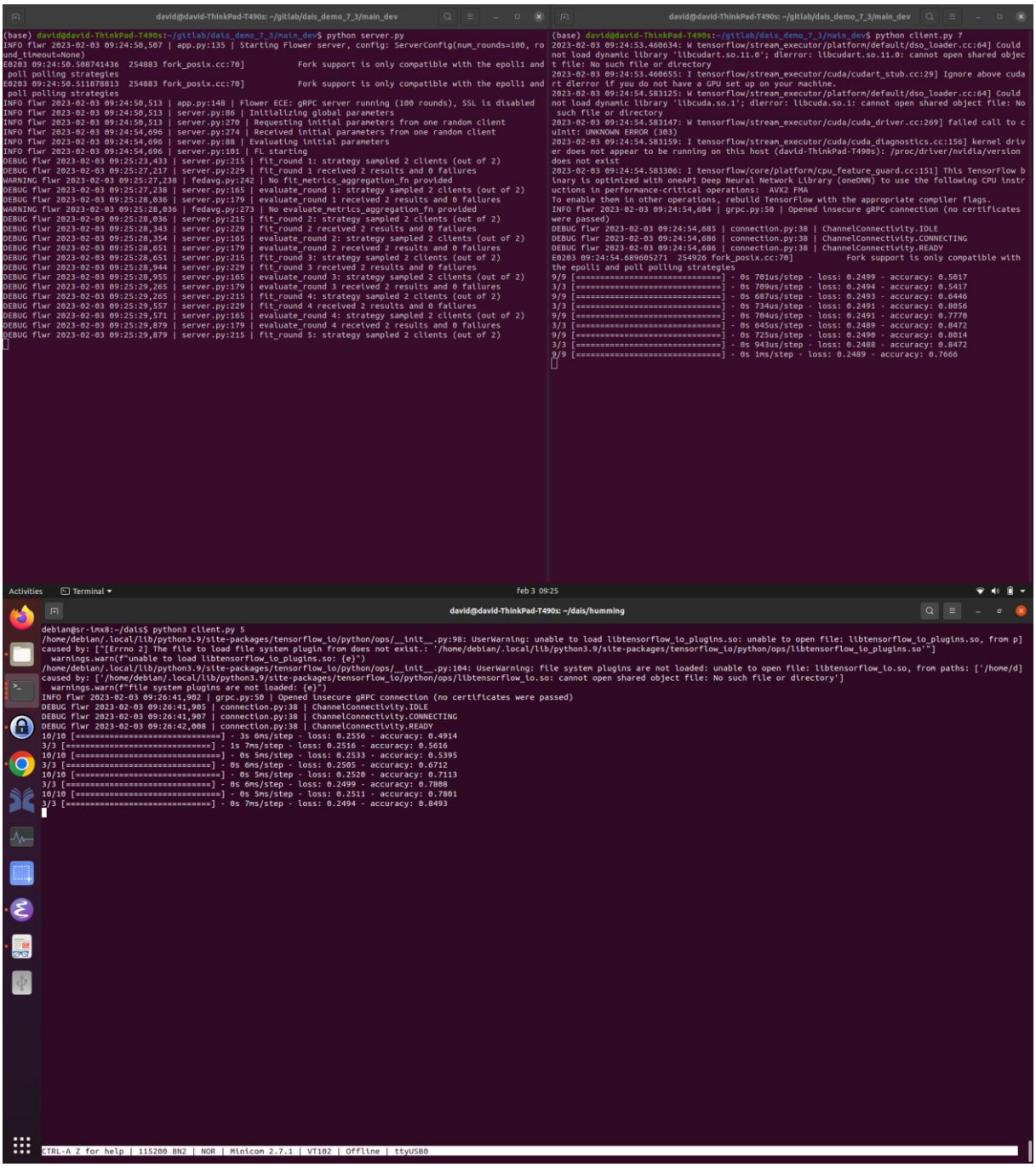
- On laptop: python server.py
- On laptop: python client.py 7
- On HummingBoard: python3 client.py 5

The arguments 7 and 5 to client.py specifies which datasets to use on the respective clients.

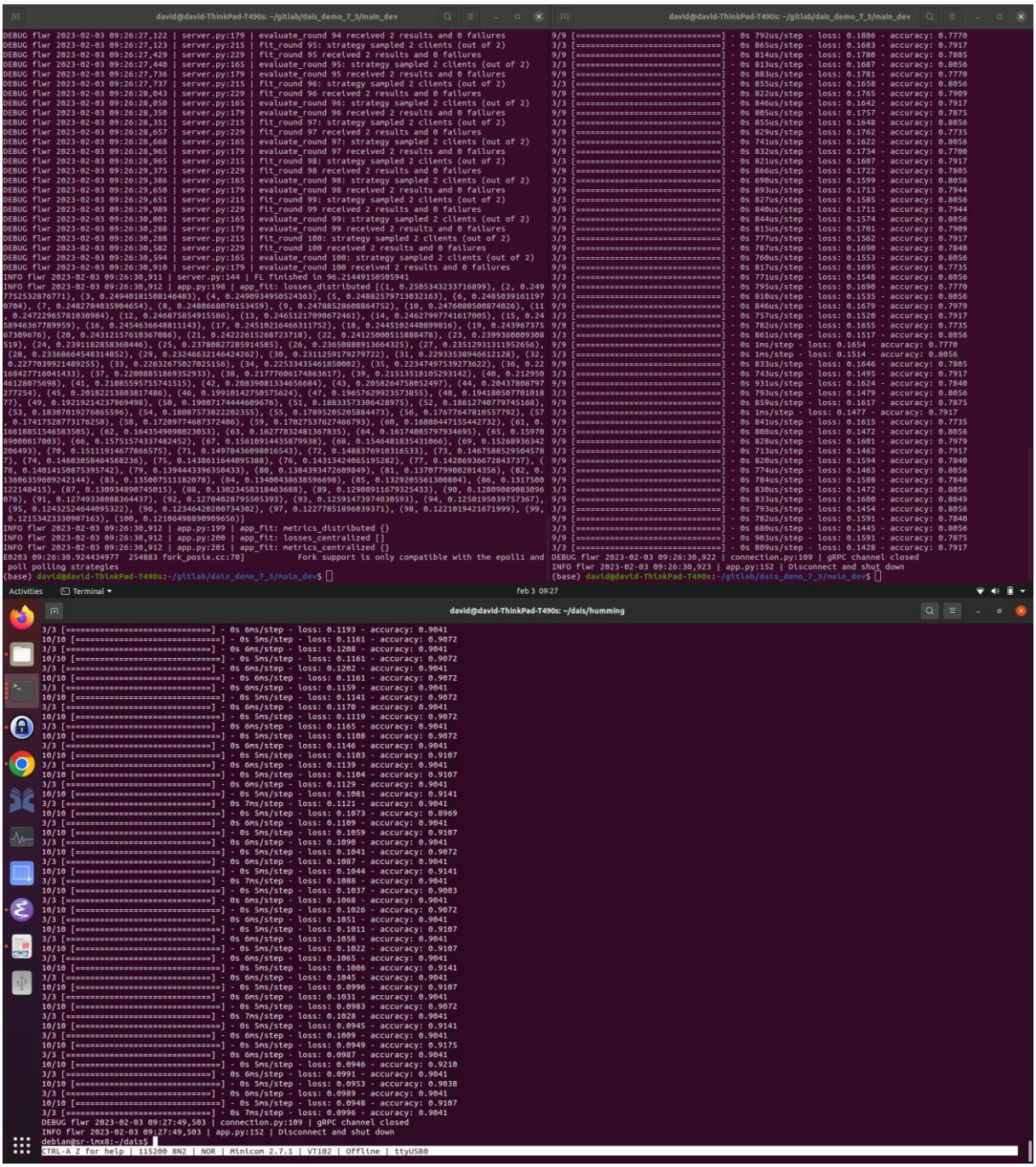
Snapshot right before the start of the second client:



Snapshot right after the start of the second client:



Snapshot at the end:



See accompanying video screencast of the training process.

Wifi connection

To connect to the local network or the internet using Wifi I did the following.

Setting up and configuring:

- Install network-tools, wpasupplicant and dhcpcd5
- Create file `/etc/network/interfaces.d/wlan0` with content:

```
allow-hotplug wlan0
iface wlan0 inet dhcp
wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf
```

```
iface default inet dhcp
```

- Create config file `/etc/wpa_supplicant/wpa_supplicant.conf` containing:

```
ctrl_interface=/run/wpa_supplicant
update_config=1

network={
    ssid="SSID"
    psk="passwd"
}
```

where SSID is the name of your network and passwd is your password. To hash password see documentation of wpasupplicant.

Connecting:

- sudo wpa_supplicant -B -i wlan0 -c /etc/wpa_supplicant/wpa_supplicant.conf
- sudo dhcpcd wlan0

Test the connection with any of the following:

- sudo apt-get update
- ping -c 3 google.com

If you are connected with ethernet and want to test your wifi connection, you can do

- ping -c 3 -I wlan0 google.com

If you are on a network without password you can do

- Create config file `/etc/wpa_supplicant/wpa_supplicant.conf` containing:

```
ctrl_interface=/run/wpa_supplicant
update_config=1

network={
    ssid="SSID"
    key_mgmt=NONE
}
```

where SSID is the name of your network

Setting up the Basler Camera

I essentially followed the instructions in the quick guide. For this section you need the above item and in addition the Camera Kit. The camera is plugged in as shown in the image below.

To use the camera I installed a Yocto image on another micro SD card as follows:

- wget https://solid-run-images.sos-de-fra-1.exo.io/IMX8/imx8mp_yocto_hardknott-5.10.72-2.2.0/imx-image-full-imx8mpsolidrun-20220216104419.rootfs.wic.xz
- xz -d imx-image-full-imx8mpsolidrun-20220216104419.rootfs.wic.xz
- Check where the SD card is mounted with fdisk -l. You will see a similar size device mounted at /dev/sdb or similar. There may be partitions at /dev/sdb1 and so on. My SD card was mounted as /dev/sdb and a partition at /dev/sdb1.
- Unmount partition with umount /dev/sdb1.
- Be sure to get the right disk in the command below (/dev/sdb in my case). The operation may erase data on your other drives if you get it wrong.
- sudo dd of=/dev/sdb if=imx-image-full-imx8mpsolidrun-20220216104419.rootfs.wic bs=4k conv=fdatasync status=progress

Then use the serial connection as above and login with username root (no password required). Now list your devices by doing

- `v4l2-ctl --list-devices`

you should find /dev/video0 or similar. To test the camera, plug a hmdi cable from the card to a monitor and do

- gst-launch-1.0 -v v4l2src device=/dev/video0 ! "video/x-raw,format=YUY2,width=1920,height=1080" ! queue ! imxvideoconvert_g2d ! waylandsink