HitSeed

Machine Learning CASE DEBARE

A SENSOR COMPUTER CASE STUDY

Collecting sensor data for training Neural Network classifiers and running embedded classifiers for real-time sensor data.

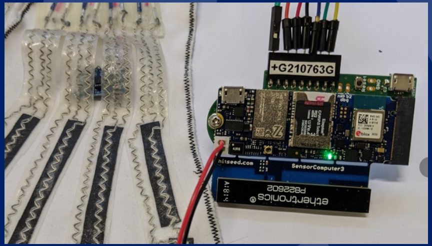
SEPTEMBER 2020

The HitSeed Sensor Computers can collect and transfer sensor data for Neural Network training and for executing the trained Neural Network in the device for embedded classifications.

This presentation demonstrates how the DEBARE smart glove classifier was set up and developed using the HitSeed Sensor Computer.

Project DEBARE

- A Horizon 2020 ATTRACT project with Aalto University https://attract-eu.com/showroom/project/deep-learning-based-activity-recognition-on-the-edge-debare/
- HitSeed Sensor Computer attached to a Smart Glove prototype developed in Aalto
- Runs neural network classifiers for recognising gestures and human activities in the glove
- Uses embedded TensorFlow Lite Micro
- Sends gesture classification results over Bluetooth LE and/or a LTE cellular data connection to a nearby computer, cloud server or VR/AR headset

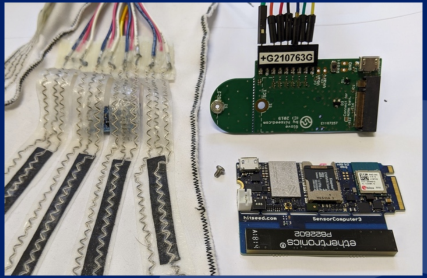


Attaching the Glove to the Sensor Computer

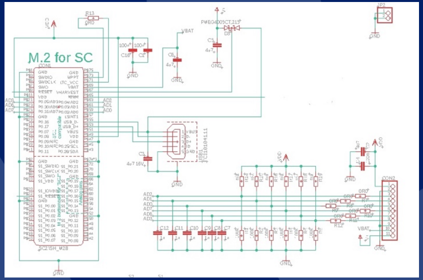
- A small Printed Circuit Board was developed for connecting the Glove and the Sensor Computer
- The board schematics on the right add a measurement front end to the glove sensors, a USB connector for charging the Li-Po battery and soldering pads for an optional battery (e.g. 2x AAA) container
- The intelligence resides in the Sensor Computer board

Aalto glove prototype

DEBARE board



Sensor Computer (SC2)



DEBARE board schematics

HitSeed

Sensors Used in DEBARE

6 smart textile sensors

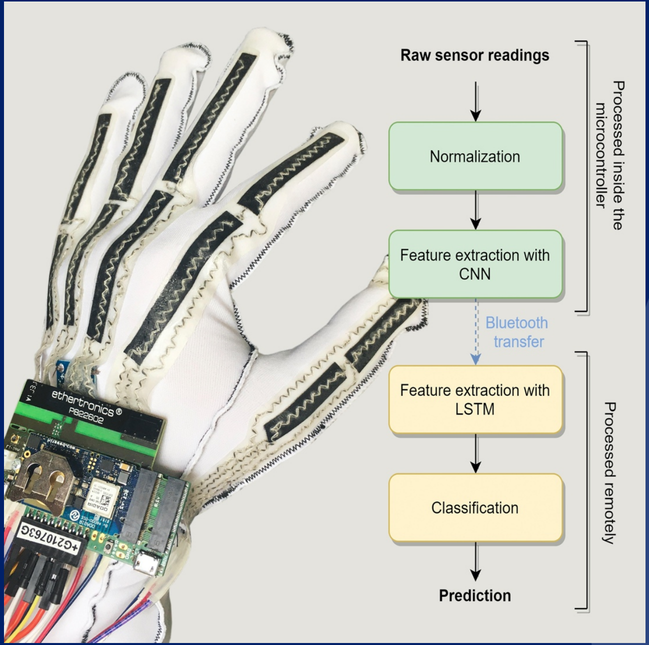
- One for each finger measuring the stretch: how much the finger has bent
- One for palm, measuring pressure
- The Sensor Computer powers these sensors and measures the varying resistance as a voltage value
- These sensors are red at 100 Hz sampling frequency (configurable 1 Hz – 1000 Hz)

Advanced 6-axis motion sensor

- Used for hand orientation and motion gesture detection
- Accelerometer and gyroscope data from the sensor is red at 104 Hz sampling frequency (configurable 1.6 Hz – 3333 Hz)

HitSeed

Steps of Classification

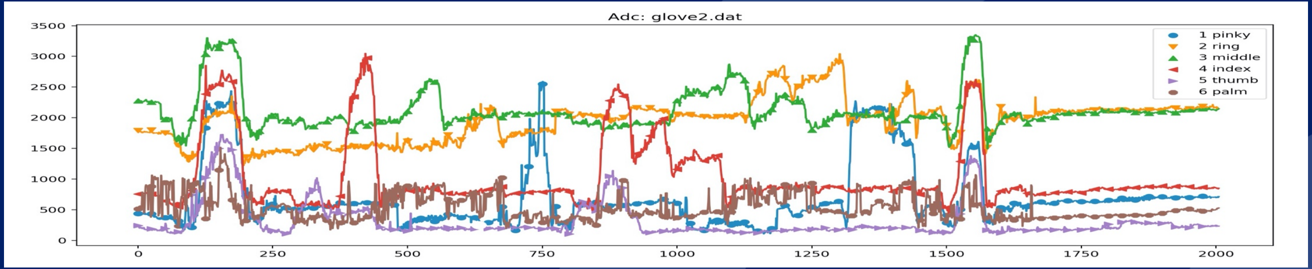


1. Configure sensors for sampling and set up filtering.
2. Set data scaling and normalisation parameters.
3. Combine several sensor inputs to a time series buffer of values for Neural Network processing.
4. Load a TensorFlow model and feed the data buffers for embedded Neural Network classification:
   1. Run Neural Network inference. Output is a list of probabilities for the trained classifier alternatives.
   2. Format the results and send over Bluetooth LE or LTE data connections. Data is sent to a server, a nearby computer or a VR/AR headset.
5. Alternatively collect the buffer data for offline supervised learning of a Neural Network classifier. Training data can be stored in a microSD memory card or sent over the LTE connection.

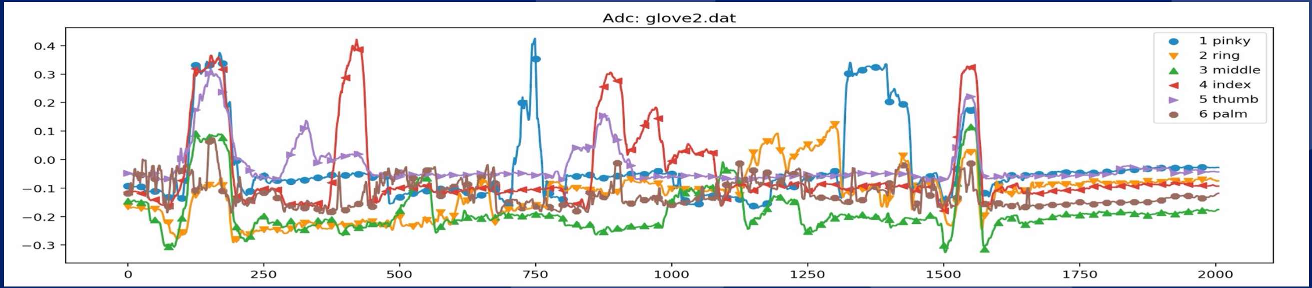
HitSeed

Step 1: Embedded Data Normalisation

Textile sensor measurements: bending all fingers first, then one or two fingers at a time.

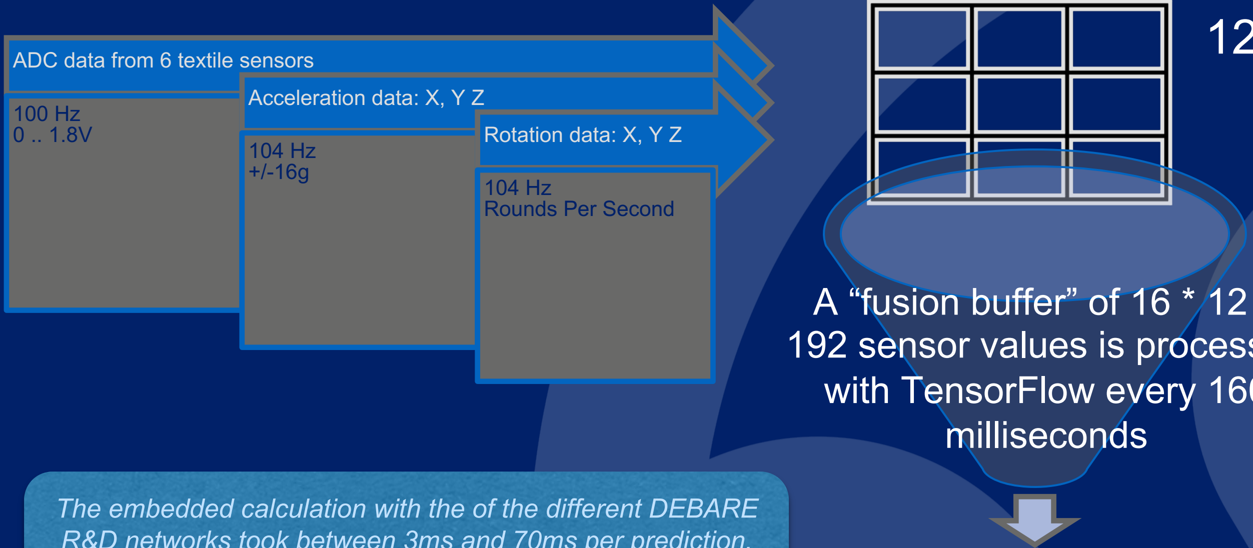


Above: raw sensor data. Below: pre-processed and normalised for Neural Network processing.



HitSeed

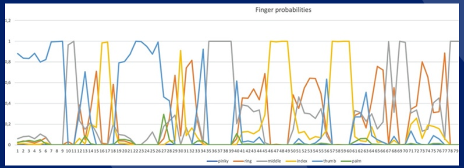
Sensor Data Fusion →TensorFlow → Result



A “fusion buffer” of 16 * 12 = 192 sensor values is processed with TensorFlow every 160 milliseconds

The embedded calculation with the of the different DEBARE R&D networks took between 3ms and 70ms per prediction. The network model of these example graphs uses on average 3.5 mW in continuous prediction of real-time data.

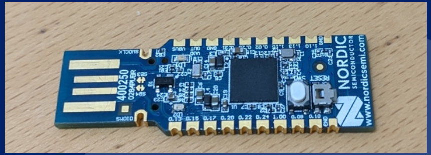
Classifier result: probabilities of each pose or gesture



HitSeed

Access the Embedded Results from a PC

- The DEBARE Sensor Computer sends a classifier result to Bluetooth LE every 160 milliseconds.
- Reliable and compatible (Windows, Mac, Linux) way to read the Bluetooth data is to use a €10 USB dongle with Python scripts. Details and code available from https://sc.hitseed.com.
- The Python script can
  - read the data from Bluetooth,
  - print the result in real-time,
  - store the values to a .csv file or
  - pass the data for processing the result with another TensorFlow network (typically a LSTM) and print the results.



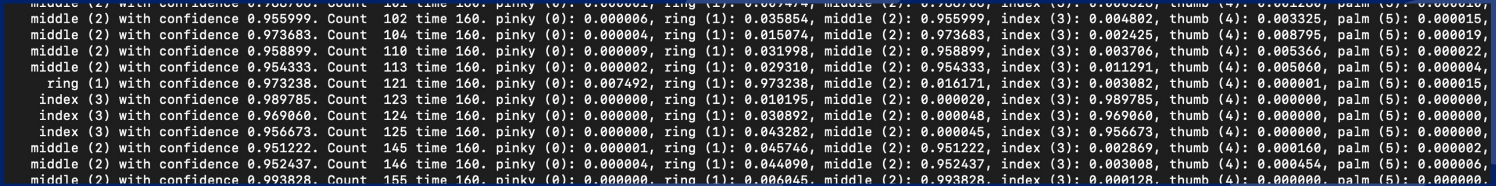
```
python3 dongle/deb_output.py
     --port=/dev/tty.usbmodemE8D77E923BE52
     --verbose
     --confidence=0.95

python3 dongle/deb_output.py
     --port=/dev/tty.usbmodemE8D77E923BE52
     --floats=6
     --floats_from_byte=6
```

HitSeed

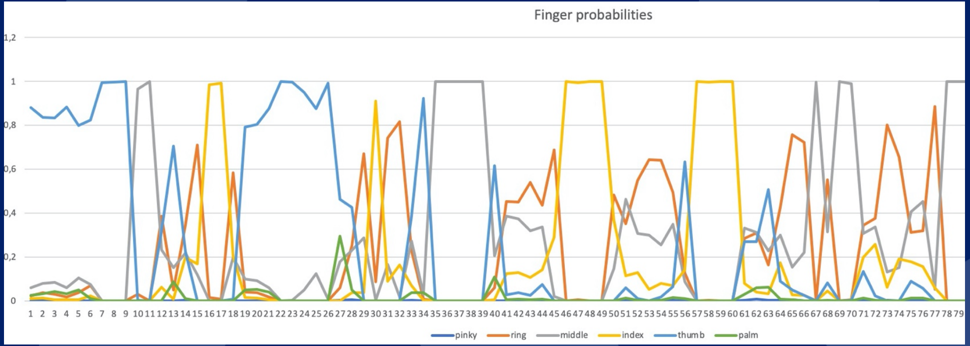
Classifier Results

The python script uses the USB dongle to receive the embedded classification as a list of probabilities for each class. This Neural Network detected which finger(s) in the glove were bent:



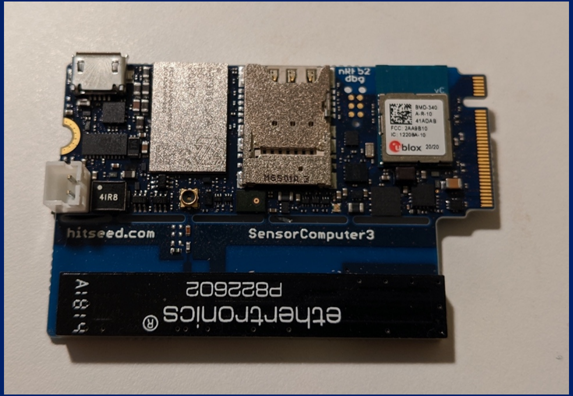
```
middle (2) with confidence 0.988766. Count  101 time 160. pinky (0): 0.000001, ring (1): 0.009474, middle (2): 0.988766, index (3): 0.000528, thumb (4): 0.001230, palm (5): 0.000010,
middle (2) with confidence 0.955999. Count  102 time 160. pinky (0): 0.000006, ring (1): 0.035854, middle (2): 0.955999, index (3): 0.004802, thumb (4): 0.003325, palm (5): 0.000015,
middle (2) with confidence 0.973683. Count  104 time 160. pinky (0): 0.000004, ring (1): 0.015074, middle (2): 0.973683, index (3): 0.002425, thumb (4): 0.008795, palm (5): 0.000019,
middle (2) with confidence 0.958899. Count  110 time 160. pinky (0): 0.000009, ring (1): 0.031998, middle (2): 0.958899, index (3): 0.003706, thumb (4): 0.005366, palm (5): 0.000022,
middle (2) with confidence 0.954333. Count  113 time 160. pinky (0): 0.000002, ring (1): 0.029310, middle (2): 0.954333, index (3): 0.011291, thumb (4): 0.005060, palm (5): 0.000004,
  ring (1) with confidence 0.973238. Count  121 time 160. pinky (0): 0.007492, ring (1): 0.973238, middle (2): 0.016171, index (3): 0.003082, thumb (4): 0.000001, palm (5): 0.000015,
 index (3) with confidence 0.989785. Count  123 time 160. pinky (0): 0.000000, ring (1): 0.010195, middle (2): 0.000020, index (3): 0.989785, thumb (4): 0.000000, palm (5): 0.000000,
 index (3) with confidence 0.969060. Count  124 time 160. pinky (0): 0.000000, ring (1): 0.030892, middle (2): 0.000048, index (3): 0.969060, thumb (4): 0.000000, palm (5): 0.000000,
 index (3) with confidence 0.956673. Count  125 time 160. pinky (0): 0.000000, ring (1): 0.043282, middle (2): 0.000045, index (3): 0.956673, thumb (4): 0.000000, palm (5): 0.000000,
middle (2) with confidence 0.951222. Count  145 time 160. pinky (0): 0.000001, ring (1): 0.045746, middle (2): 0.951222, index (3): 0.002869, thumb (4): 0.000160, palm (5): 0.000002,
middle (2) with confidence 0.952437. Count  146 time 160. pinky (0): 0.000004, ring (1): 0.044090, middle (2): 0.952437, index (3): 0.003008, thumb (4): 0.000454, palm (5): 0.000006,
middle (2) with confidence 0.993828. Count  155 time 160. pinky (0): 0.000000, ring (1): 0.006045, middle (2): 0.993828, index (3): 0.000128, thumb (4): 0.000000, palm (5): 0.000000,
```

The 160 millisecond predictions as a graph:



HitSeed

DEBARE Hardware Uses Sensor Computer 2



Sensor Computer 2 (SC2) is the main product of HitSeed: the IoT "brains" used in several commercial products.

SC2 contains a microphone and an advanced motion sensor. It is often attached to another board for added sensors and actuators. SC2 runs the software that measures and sends data, it has memory, non-volatile storage, programmability and it communicates with Bluetooth LE and with LTE (4G) NB-IOT/CAT-M1.

SC2 weights 10 grams. It has several powering options including primary and rechargeable batteries and energy harvesting. The DEBARE device in idle mode uses only 304 µW – as a comparison a Raspberry Pi Zero uses 400 mW in idle mode.

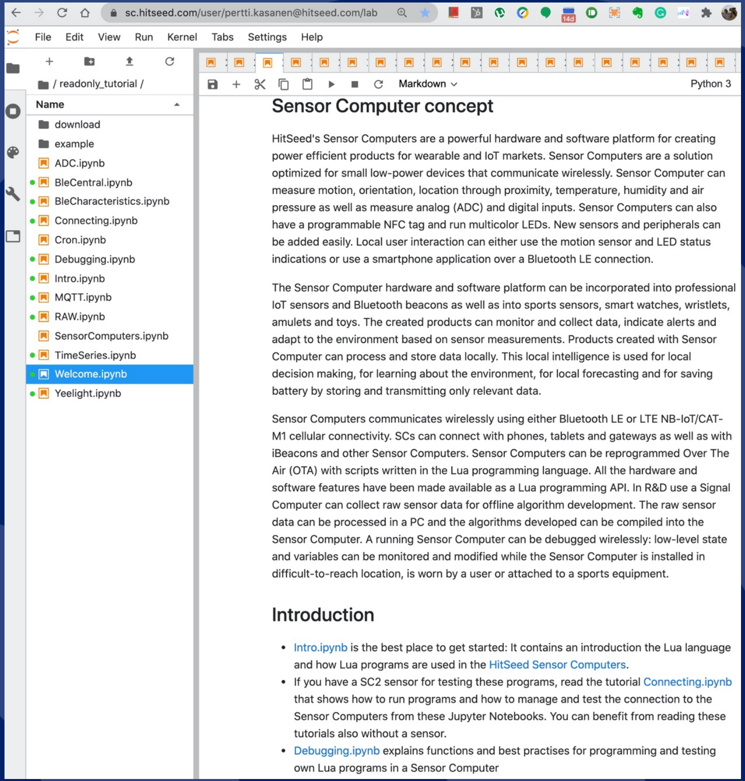
There is a smaller and lighter variant Sensor Computer 1 (SC1) available for a smart glove in production, without the LTE data connectivity. For R&D and for training data collection the larger SC2 variant with a memory card holder has been easier to use.

HitSeed

Programming the Sensor Computers

The HitSeed Sensor Computers are configured using the Lua script language. The Lua programs can be updated to the device over the air.

Tutorials of the Sensor Computer programming are available in https://sc.hitseed.com.



Sensor Computer concept

HitSeed's Sensor Computers are a powerful hardware and software platform for creating power efficient products for wearable and IoT markets. Sensor Computers are a solution optimized for small low-power devices that communicate wirelessly. Sensor Computer can measure motion, orientation, location through proximity, temperature, humidity and air pressure as well as measure analog (ADC) and digital inputs. Sensor Computers can also have a programmable NFC tag and run multicolor LEDs. New sensors and peripherals can be added easily. Local user interaction can either use the motion sensor and LED status indications or use a smartphone application over a Bluetooth LE connection.

The Sensor Computer hardware and software platform can be incorporated into professional IoT sensors and Bluetooth beacons as well as into sports sensors, smart watches, wristlets, amulets and toys. The created products can monitor and collect data, indicate alerts and adapt to the environment based on sensor measurements. Products created with Sensor Computer can process and store data locally. This local intelligence is used for local decision making, for learning about the environment, for local forecasting and for saving battery by storing and transmitting only relevant data.

Sensor Computers communicates wirelessly using either Bluetooth LE or LTE NB-IoT/CAT-M1 cellular connectivity. SCs can connect with phones, tablets and gateways as well as with iBeacons and other Sensor Computers. Sensor Computers can be reprogrammed Over The Air (OTA) with scripts written in the Lua programming language. All the hardware and software features have been made available as a Lua programming API. In R&D use a Signal Computer can collect raw sensor data for offline algorithm development. The raw sensor data can be processed in a PC and the algorithms developed can be compiled into the Sensor Computer. A running Sensor Computer can be debugged wirelessly: low-level state and variables can be monitored and modified while the Sensor Computer is installed in difficult-to-reach location, is worn by a user or attached to a sports equipment.

Introduction

- Intro.ipynb is the best place to get started: It contains an introduction the Lua language and how Lua programs are used in the HitSeed Sensor Computers.
- If you have a SC2 sensor for testing these programs, read the tutorial Connecting.ipynb that shows how to run programs and how to manage and test the connection to the Sensor Computers from these Jupyter Notebooks. You can benefit from reading these tutorials also without a sensor.
- Debugging.ipynb explains functions and best practises for programming and testing own Lua programs in a Sensor Computer

Training a TensorFlow .tflite Model

To collect the training data files and to use the files to train a TensorFlow model you should follow the steps in the next few slides.

HitSeed

Outline a typical development of a Machine Learning classifier

1. Training data collection with a SC

- Collect a few 10 - 60 second activity samples of each activity, gesture or state
- Collect minimum 5 sample files of each gesture to be trained

2. Neural Network training

- Use HitSeed-provided tools to process the training data
- Train a Tensor Flow Lite model in a reasonably fast PC or server

3. Run classifier in a Sensor Computer

- Upload the trained .tflite model to a Sensor Computer
- Create a script program that runs in the Sensor Computer and
  - Passes sensor data to the TFL model for classification
  - Formats and sends the classifier results

HitSeed

Training Preparations

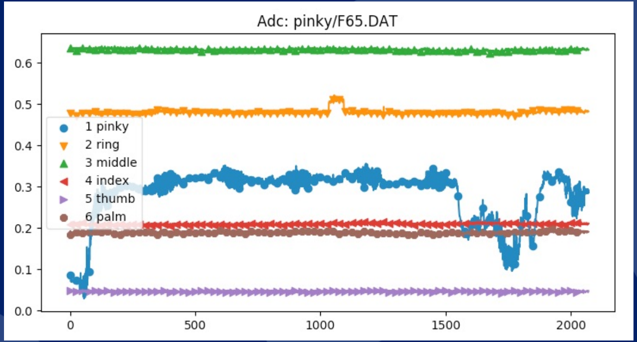
1. To install a Python interpreter with Tensorflow and Keras follow the instructions from tensorflow.org
2. Login to http://sc.hitseed.com to download the training scripts
3. More detailed instructions for NN training and for Lua programming with Sensor Computers are available in http://sc.hitseed.com

HitSeed

Training: Inspect and Clean Up Data

- Copy the training data files from a microSD to your computer. Place the files in subdirectories named as the pose or gesture to be classified.
- You can later choose which subdirectories to include in each training run.
- Inspect the files one by one by plotting the contents with commands like on the right.
- Verify what data you want to include in the training. Move misplaced files to correct subdirectories and remove any undesired files. The script can also remove samples from the beginning and from the end.

```
python3 fusion_parse.py */* --plot_adc
python3 fusion_parse.py */* --plot_acc
python3 fusion_parse.py */* --plot_gyro
```



HitSeed

Training: Prepare Data for Training

- Convert the verified fusion buffer binary files as .csv files in the directory specified as the `--csv_to_dir` argument.
- If needed, split the files with `--csv_split=<numberoflines>` to get at least ten .csv files for each pose/gesture.

- Run these training data preparation scripts. List the subdirectory names you want to include in this training. The order of the directories has no impact here.
- Rerun these last two commands for changing the subdirectories included for training.

```
python3 fusion_parse.py */* --csv_to_dir=../csv
    --csv_split=1000


cd ../csv

python3 ../../train/data_prepare.py --inputs=12
index middle palm pinky ring thumb

python3 ../../train/data_split.py index middle
palm pinky ring thumb
```

HitSeed

Training: Run the TensorFlow Training

- Use the train.py script from http://sc.hitseed.com for TensorFlow training.
- train.py uses the learning/validation/test data prepared with `data_split.py`
- Please refer to Tensorflow documentation about the parameters like learning rate, epochs and dropout parameters.
- In the command line list the same subdirectories for training as in the previous page. The order of the category names here defines the category order in the output.
- As a starting point we recommend using Keras for describing the network to train. It is possible to use also all other forms that TensorFlow can convert to a .tflite model.
- When the TensorFlow training run is successful it will store in the current working directory a .tflite file optimised for embedded execution.

```
python3 ../train.py
    --model=CNN_ADC
    --samples=16
    --cnn_nodes=6
    --dropout_input=0.2
    --dropout_hidden=0.0
    --learning_rate=0.001
    --steps_per_epoch=5000
    --epoch=20
    pinky ring middle index thumb palm
```

HitSeed

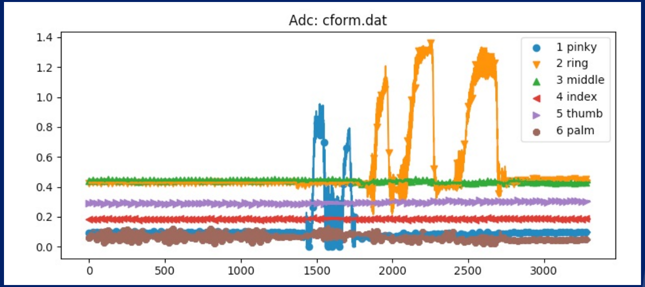
Easy Start Without Embedded Programming

HitSeed creates ready made data aggregations designed for classifier training and execution.

We call this feature **classformation** (CFORM): *Information about the sensor data aggregated to a form that is optimised for classifier training.*

CFORM Example from the DEBARE Data

These graphs illustrate only the ADC "finger" part of the DEBARE data

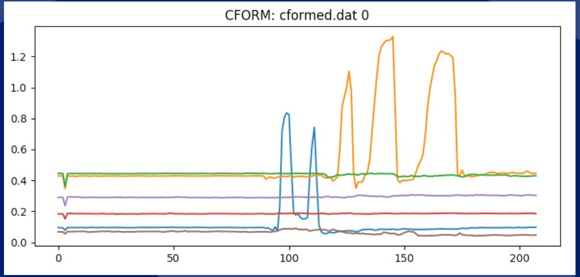


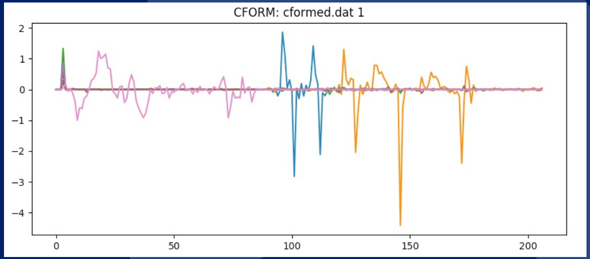
TensorFlow
(1,16,12,1)

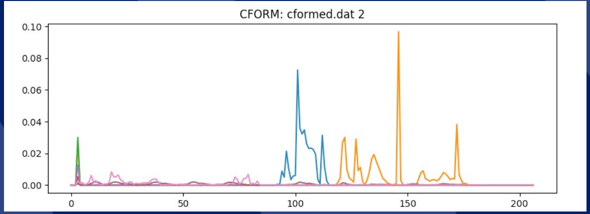
Reduction to
3/16 = 19% of the
original data size

CFORM

TensorFlow
(1,1,12,3)







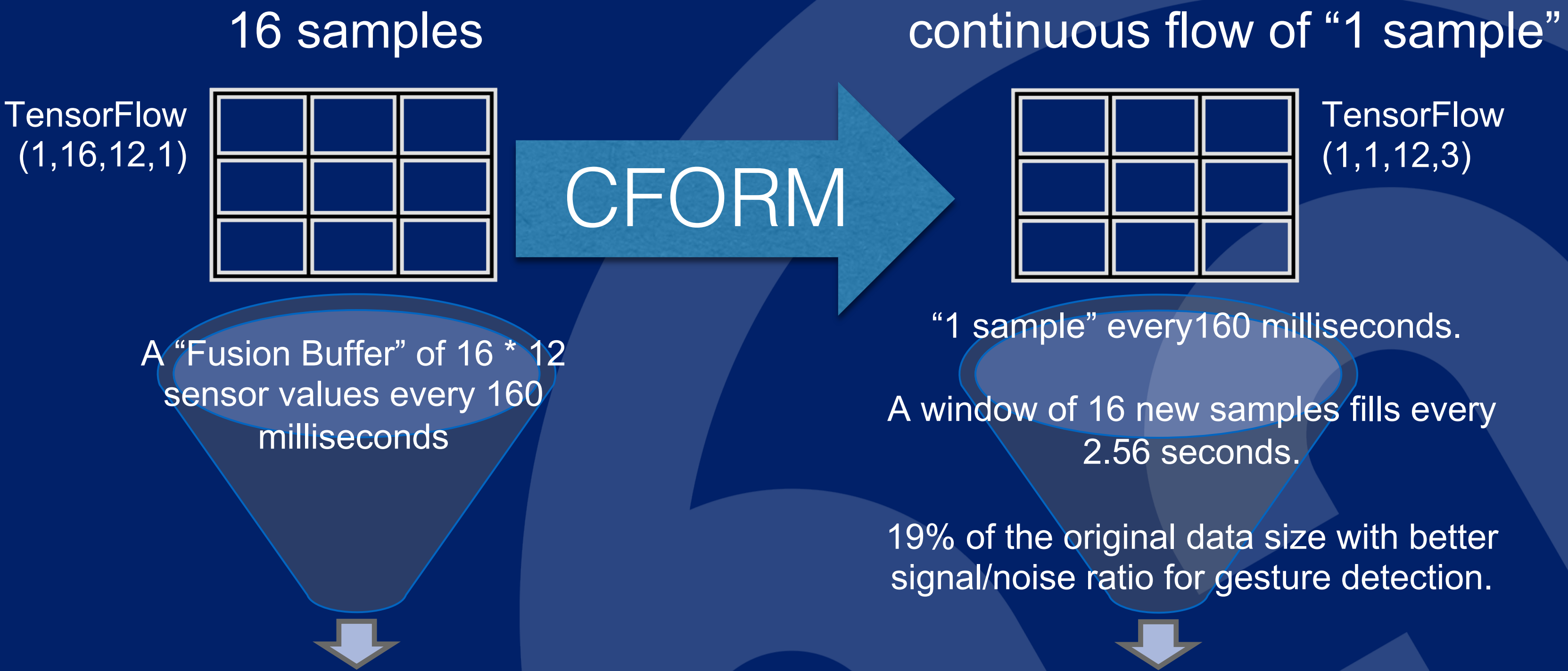
HitSeed

Classformation for Gestures and Poses

- The first CFORM feature CFORM/Gesture is developed for classifiers of dynamic human gestures and static poses.
- We take 16 samples of sensor data and calculate three numbers that represent the same information in more compact form.
    - The aggregate data is 19% in size and compact enough to be transferred over Bluetooth LE connection in real-time.
    - The new CFORM data can be saved to a microSD card for training data collection. Alternatively the training data files can be collected in real-time in a PC with a USB dongle.
    - A separate Neural Network classifier can be run in a PC or a mobile phone that consumes the real-time CFORM data sent from a Sensor Computer over Bluetooth LE.

HitSeed

CFORM/Gesture Data Visualised



A “Fusion Buffer” of 16 * 12 sensor values every 160 milliseconds

“1 sample” every160 milliseconds.

A window of 16 new samples fills every 2.56 seconds.

19% of the original data size with better signal/noise ratio for gesture detection.

Usually better classifier results from CFORM data, with less memory and power consumption.

HitSeed

Training Classifiers for the CFORM/Gestures

- Easiest way to start training classifiers with HitSeed Sensor Computers is to take a SC configured for sending the CFORM/Gestures data to Bluetooth.
  - Use a USB dongle and Python scripts to collect CFORM training data to a PC.
  - Train a TensroFlow classifier using the collected compact CFORM data.
  - Load the trained TensorFlow model with the dongle-reader Python script running in the PC and test the NN model against the real-time data received over Bluetooth.

- The current version of the TensorFlow Lite Micro cannot yet run LSTM networks. When RNN-type networks or subgraphs are used for in classification, currently this kind of split setup is needed where the more complex classifier runs in a PC or a mobile phone.

HitSeed

Embedded Use of CFORM/Gestures

- The CFORM outputs can be sent for embedded TensorFlow inference from the Lua code. Up to 2.56 second gestures could be classified with a 16-sample CNN in the Sensor Computer.
- The CFORM transformation helps reducing the memory footprint of the classifier model as well as the RAM memory needed for TensorFlow Arena. The calculation speed and the power consumption improves – allowing more complex embedded NN topologies.
- If the trained NN uses only a fully connected or a CNN network and has <3000 trainable variables, it can be downloaded to the Sensor Computer and run the classifications fully locally.
- The CFORM model makes it easy to start training classifiers with full TensorFlow functionality – and the gradually trim the complexity of the model to find out how many separate gestures can be reliably classified in embedded-only implementation.

- www.hitseed.com/platform
- info@hitseed.com

HITSEED.COM