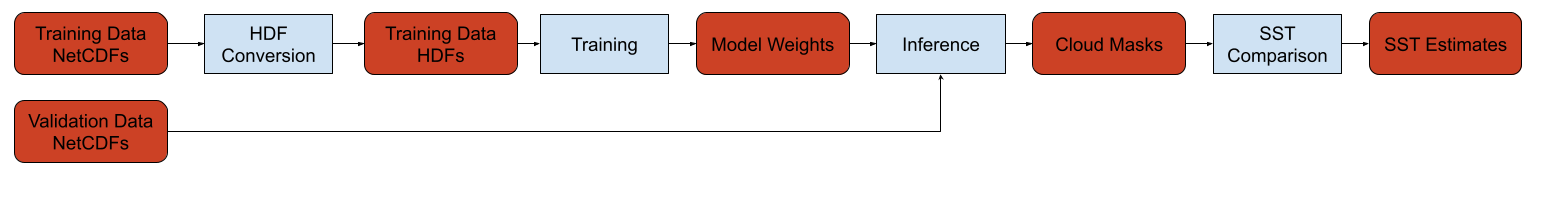
# Case Two: Cloud Benchmark

This report provides a brief overview of case two of the intel end-to-end benchmarking project.

## Pipeline Components

Diagram of the full end-to-end workflow of the benchmark. Red components indicate the artifacts input or output from each stage. Blue components indicate a step in the processing workflow.



The following sections will provide a brief description of how to run the benchmark components.

## Software

The benchmark is written in pure python code. The preprocessing scripts make use of the NetCDF, H5py, and scikit-image libraries pre-processing the imagery. The network uses a U-Net style architecture with 9 channels as input (6 channels reflectance, 3 channels brightness temperature) and a single channel binary output. This network is written in Tensorflow 2.0. A full list of software requirements can be found in the [requirements.txt](https://github.com/stfc-sciml/e2e/blob/master/case2/requirements.txt) in the github repository.

### Data Extraction

The extraction step unzips the raw data from the CEDA archive. This only needed to be performed once for each data product. It is not necessary to perform this step to execute the benchmark stages as the data have already been extracted and added to a public data repository. (see datasets below). The extraction step takes an input list of file locations on the CEDA archive and will unzip every file in the list to the corresponding output directory. An example of how to run this step is given below:

*python -m e2e\_benchmark.command extract file\_list.txt extracted\_files*

### HDF Conversion

The HDF conversion converts the raw NetCDF files into a stand alone HDF file. This collects all of the brightness temperature channels, radiance channels, and product masks into a single file. This step is necessary to cut down the number of I/O calls during training. This step also converts the radiance channels to reflectance values and resizes the images to a common size. Finally, the data will be split into day and night time image folders. An example of how to run this step is given below:

*python -m e2e\_benchmark.command convert\_hdf extracted\_files hdf\_files*

### Training

To train the model you will need to have performed both preprocessing tasks and should now have a folder to HDF files ready. The input parameters for the model training are the folder of HDF files and an output path to save the model to. This will run the U-net like model with each of the images in the input folder. The input images will be automatically split into a training & test set.

*python -m e2e\_benchmark.command train hdf\_files model\_output*

To run the model in CPU only mode you can pass the additional flag --cpu-only.

As part of a the data loading the following operations will be performed:

* Image normalisation of each channel
* Conversion from full (1500x1200) resolution to patches (512x512).
* Cache these operations in memory using tf.data.Dataset.cache

### Inference

After the model has been trained you can run it on a set of test images using the inference command. The inference command takes three arguments:

* The model file containing a trained model.
* The data directory containing preprocessed HDF data to perform inference on
* An output directory to predicted masks to for each image in the data directory

An example of how to run this step is given below:

*python -m e2e\_benchmark.command inference model\_output/model\_file.h5 hdf\_files predictions*

### SST Comparison

The final step of the benchmark is to compare the masked pixels in the validation SST dataset against the reference SST temperature buoys. For this you will specifically need the preprocessed ssts dataset. This step will take the table of SST match ups and compare the masks output from the model with those match ups and output the median and robust standard deviation for the match ups. An example of how to run this step is given below:

*python -m e2e\_benchmark.command sst\_comp sst\_matchups.h5 predictions*

## Datasets

Three datasets are provided with this benchmark. The details of each individual dataset are given in the table below. Each dataset contains multiple folders referred to as level 1 Sentinel-3 SLSTR products. Each product contains multiple data files containing the raw brightness temperatures and radiances measured by the satellite. For more details on the specific contains of the level 1 products, the reader is referred to the [SLSTR level 1 handbook](https://sentinel.esa.int/documents/247904/1872792/Sentinel-3-SLSTR-Product-Data-Format-Specification-Level-1).

Each of these datasets are publicly available in an S3 storage bucket. A [download script](https://github.com/stfc-sciml/e2e/blob/master/download.sh) has been provided inside the git repository. The benchmark download script requires aws-shell to be installed and available on the PATH. To install aws-shell run:

pip install aws-shell

To download benchmark data run:

*./download.sh <output-folder>*

For example:

*./download.sh data*

Some details of the provided datasets are listed below:

|  |  |  |  |
| --- | --- | --- | --- |
| Dataset | No. Products | Size (GB) | Description |
| one-day | 971 | 599 | Data recorded during a single full orbit cycle of Sentinel-3A. Consists of day and night time examples and images from a wide variety of biomes. |
| pixbox | 414 | 243 | A mix of data used as part of a cloud mask validation dataset containing numerous difficult examples across a variety of biomes. |
| ssts | 100 | 63 | A dataset of products which have been co-aligned with sea surface temperature (SST) buoys. The matching pixels in these products can be used to validate the quality of cloud masking |

In addition to each of these raw datasets, the data transformed preprocessed to the HDF format is also provided. This is the intermediate output after running convert\_to\_hdf on each of the one-day and ssts dataset. These intermediate files can be found in ./hdf and are about ~30% the size of the original dataset.

## Results

This section reports the results of running the benchmark training and inference steps on three different HPC systems. A subset of results are provided here, full result tables can be found in the GitHub repository in [here.](https://github.com/stfc-sciml/e2e/blob/master/case2/docs/results.xlsx)

### PEARL

PEARL is a high-performance computing cluster, designed primarily for Deep Learning and AI research. At its core are two [NVIDIA DGX-2s](https://www.nvidia.com/en-gb/data-center/dgx-2/). Each DGX-2 utilises 16 Tesla V100 GPUs with a total of 512GB of GPU memory. Each also has 1.5TB of system RAM, two Intel Xeon Platinum CPUs and 30TB of NVME SSD local storage. The DGX-2s are connected over 100Gbit/s EDR InfiniBand to two Boston Flash-IO Talyn servers which together provide half a petabyte of NVMe storage. For more information please see the [PEARL user documentation](https://pearl-cluster.readthedocs.io/en/latest/index.html#). The benchmark was run on both a single DGX\_2 machine and distributed across both DGX-2 nodes this system to evaluate training & inference performance.

#### *Training*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **index** | **system** | **num\_ranks** | **num\_gpus** | **total\_time** |
| run\_24409 | PEARL | 1 | 1 | 4686.985232 |
| run\_24410 | PEARL | 2 | 2 | 3681.385369 |
| run\_24411 | PEARL | 4 | 4 | 1828.20399 |
| run\_24412 | PEARL | 8 | 8 | 1014.694396 |
| run\_24413 | PEARL | 16 | 16 | 562.1420362 |

#### Inference

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **index** | **system** | **num\_gpus** | **num\_ranks** | **total** | **imgs\_per\_s** |
| run\_24409 | PEARL | 1 | 1 | 1255.247758 | 1.593310951 |
| run\_24410 | PEARL | 2 | 2 | 1256.608641 | 1.591585427 |
| run\_24411 | PEARL | 4 | 4 | 539.1232088 | 3.709727141 |
| run\_24412 | PEARL | 8 | 8 | 265.4282806 | 7.53499211 |
| run\_24413 | PEARL | 16 | 16 | 153.5484927 | 13.02520113 |

### SCARF

SCARF is a high-performance computing cluster, with 17 GPU nodes, each with 4 NVIDIA K80 GPUs. For more information please see the SCARF user documentation. The benchmark was run across multiple GPU nodes on this system to evaluate training & inference performance.

#### *Training*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **index** | **system** | **num\_gpus** | **num\_ranks** | **total\_time** |
| run\_53428 | SCARF | 1 | 1 | 45402.69838 |
| run\_53429 | SCARF | 2 | 2 | 23281.76684 |
| run\_53430 | SCARF | 4 | 4 | 12457.26462 |
| run\_53431 | SCARF | 4 | 8 | 4786.609712 |
| run\_53432 | SCARF | 4 | 16 | 1615.83125 |
| run\_53433 | SCARF | 4 | 32 | 1006.319253 |

#### Inference

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **index** | **system** | **num\_gpus** | **num\_ranks** | **total** | **imgs\_per\_s** |
| run\_53428 | SCARF | 1 | 1 | 1494.222932 | 1.338488359 |
| run\_53429 | SCARF | 2 | 2 | 1048.268264 | 1.907908566 |
| run\_53430 | SCARF | 4 | 4 | 664.9484975 | 3.007751739 |
| run\_53431 | SCARF | 4 | 8 | 297.8718445 | 6.714296892 |
| run\_53432 | SCARF | 4 | 16 | 111.2567739 | 17.97643352 |
| run\_53433 | SCARF | 4 | 32 | 78.90235639 | 25.34778544 |

### EPCC

EPCC operates a high-performance computing CPU cluster. The benchmark was run by distributing across multiple CPU nodes on the system to evaluate training & inference performance. On this machine non GPUs were used, instead performance was compared between an Intel optimized version of Python and Tensorflow and unoptimised Python and Tensorflow binaries.

#### Training

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **index** | **python** | **system** | **num\_ranks** | **ntasks\_per\_node** | **total\_time** |
| run\_66576 | vanilla | EPCC | 4 | 4 | 51649.54848 |
| run\_66577 | vanilla | EPCC | 8 | 4 | 23524.96386 |
| run\_66578 | vanilla | EPCC | 16 | 4 | 11379.68126 |
| run\_66579 | vanilla | EPCC | 32 | 4 | 8428.152408 |
| run\_66584 | intel | EPCC | 4 | 4 | 43590.22756 |
| run\_66585 | intel | EPCC | 8 | 4 | 19652.99672 |
| run\_66586 | intel | EPCC | 16 | 4 | 10048.95009 |
| run\_66587 | intel | EPCC | 32 | 4 | 5018.560455 |

#### Inference

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **index** | **system** | **python** | **num\_ranks** | **ntasks\_per\_node** | **total** | **imgs\_per\_s** |
| run\_66576 | EPCC | vanilla | 4 | 4 | 200.0939715 | 9.995303632 |
| run\_66577 | EPCC | vanilla | 8 | 4 | 107.6166244 | 18.58448926 |
| run\_66578 | EPCC | vanilla | 16 | 4 | 59.96940947 | 33.35033674 |
| run\_66579 | EPCC | vanilla | 32 | 4 | 31.91276693 | 62.67084281 |
| run\_66584 | EPCC | intel | 4 | 4 | 257.3960884 | 7.770125851 |
| run\_66585 | EPCC | intel | 8 | 4 | 133.802573 | 14.94739567 |
| run\_66586 | EPCC | intel | 16 | 4 | 73.51474023 | 27.20542838 |
| run\_66587 | EPCC | intel | 32 | 4 | 38.53717971 | 51.89793376 |