CERN openlab

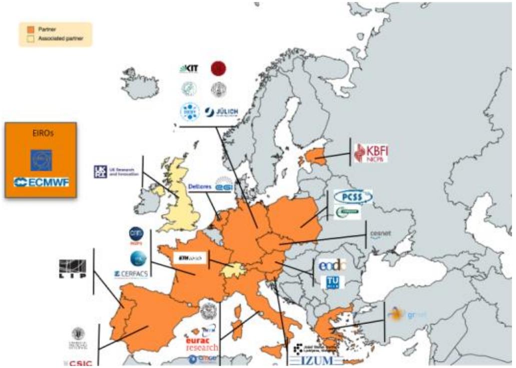
AI Methods for Digital Twins for Scientific Applications

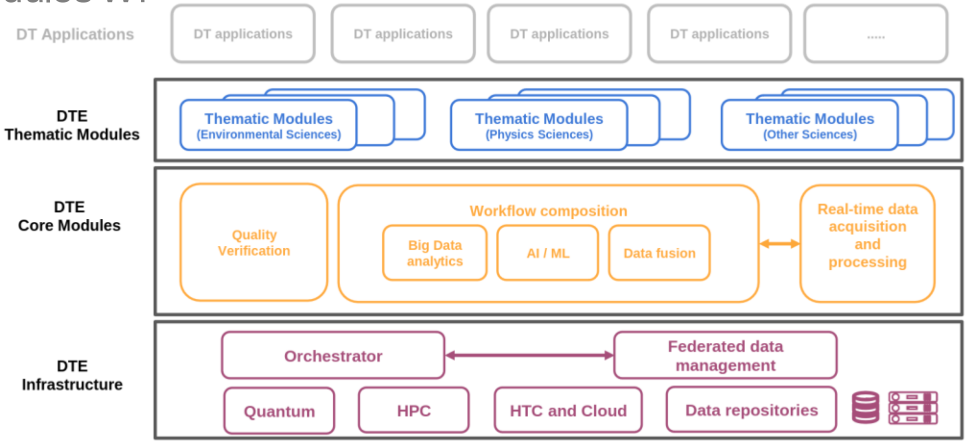
Roman Machacek
Supervisors: Matteo Bunino, Alexander Zoechbauer

CERN openlab

InterTwin project

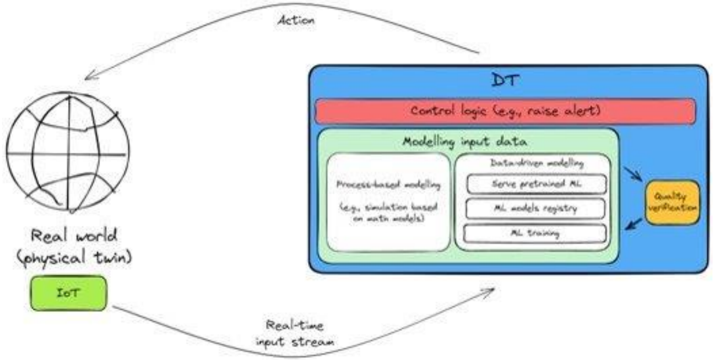
- Design and implement a prototype of an interdisciplinary Digital Twin Engine
- Different work packages (use cases, thematic modules, core modules, infrastructure)
- Work done in AI/ML task within Core modules WP





Digital Twins & AI

- What is Digital Twin? (vs Simulation)
- Reinforcement learning, ML & AI
- Many use cases, need to generalize (TF, Torch)
- Distribution and Tuning

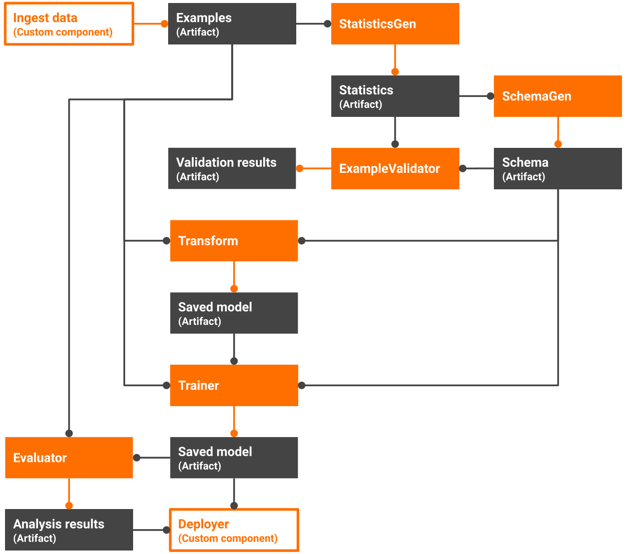


State, Goals and General Directions

- State before: CLI for MNIST, workflows, simple logging
- Goals: Abstraction, Pipelining, Distribution & Hyperparameter tuning, Logging
- New use-case: Cyclone Detection
- Where to start?

Pipelining

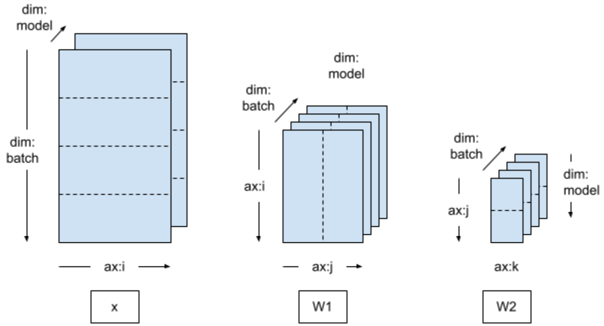
- Inspiration:
  - TFX Tensorflow Pipelines
  - Sklearn pipelines
- Abstraction:
  - Executable (Trainer, Dataloading, Processing)
  - Executor
- Step by step execution, Configurable, Generalizable



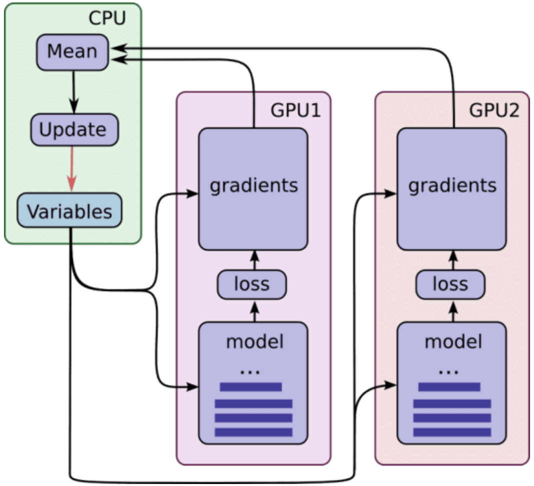
TFX pipeline. Image credits: Google, https://www.tensorflow.org/tfx/guide/understanding_tfx_pipelines

Distributed ML (Pytorch, TF)

- Different frameworks have different distributions
- Unify it in for the developer
- Distribution: Local (One node) vs Global (More nodes)
- Tensorflow (Keras): Distributed Dataset, Trainer
- Pytorch (Lightning): Distribution via ,,MPI"
- DistributedTrainer:
  - Use scope to define workers
  - execute in distributed fashion
  - New trainer simply uses super().train(data) to distribute training



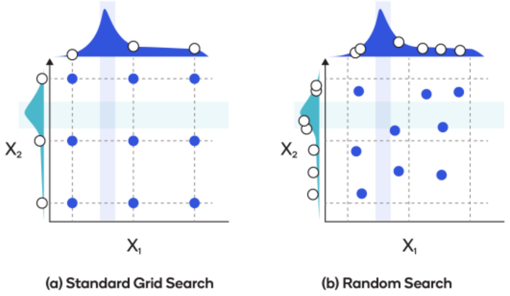
Data parallelization, Image credits: Google, https://www.tensorflow.org/tutorials/distribute/dtensor_ml_tutorial



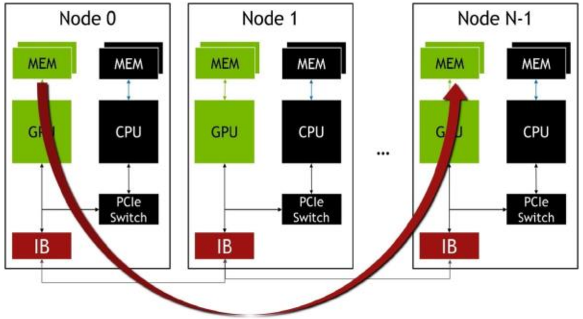
Node distribution. Image credits: Google, Aymeric Damien https://github.com/aymericdamien/TensorFlow-Examples/blob/master/tensorflow_v2/notebooks/6_Hardware/multigpu_training.ipynb

Future: Hyperparameter Optimization

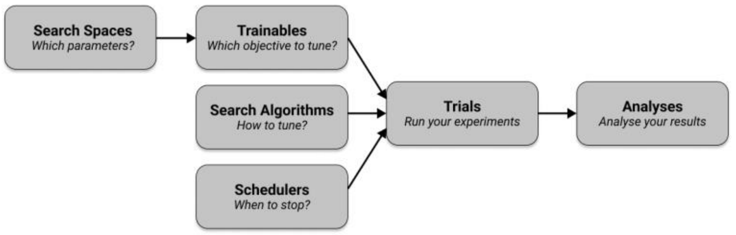
- Simplest way: Grid search, but can we do better?
- Many libraries, focus on Ray (distribution)
- One node per hyper-param config, then local distribution
- Distributed executor (run pipeline on each worker)



Search spaces. Image credits: StackOverflow, https://stackoverflow.com/questions/65682419/how-to-plot-grid-seach-layout-and-random-search-layout



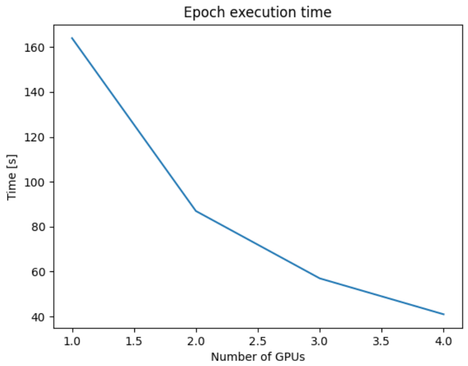
Multiple Nodes. Image credits: NVIDIA, https://www.nvidia.com/en-us/on-demand/session/gtcspring21-s31050/

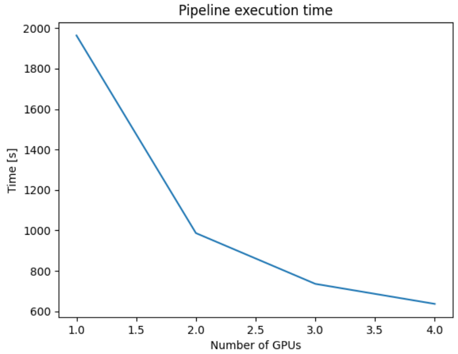


Ray pipeline. Image credits: Ray, https://docs.ray.io/en/latest/tune/key-concepts.html

State & Future directions

- State:
  - Configurable pipelines
  - Distributed model training & tuning
  - Logging (possibility for multiple loggers)
  - CycleGAN, Conv models as examples
  - Use-case integration (Cyclones)
- Visualization of pipelines
- Execution on Kubernetes
- Training achieved only through config






```yaml
executor:
  class_path: executor.CycloneExecutor
  init_args:
    run_name: 'default'

getter:
  class_path: dataloader.TensorflowDataGetter
  init_args:
    patch_type: NEAREST
    shuffle: False
    split_ratio: [0.75, 0.25]
    batch_size: 16
    augment: True
    epochs: 1
    target_scale: False
    label_no_cyclone: NONE
    aug_type: ONLY_TCS
    experiment: {
      'DRV_VARS_1': ['fg10', 'msl', 't_500', 't_300'],
      'COO_VARS_1': ['patch_cyclone'],
      'MSK_VAR_1': None
    }

trainer:
  class_path: trainer.TensorflowTrainer
  init_args:
    network: VGG_V1
    activation: LINEAR
    regularization_strength: NONE
    learning_rate: 0.0001
    loss: MAE
```

Thank you for your attention

CERN openlab