ACTS Developers Workshop 2022

Exa.TrkX & ACTS tutorial session

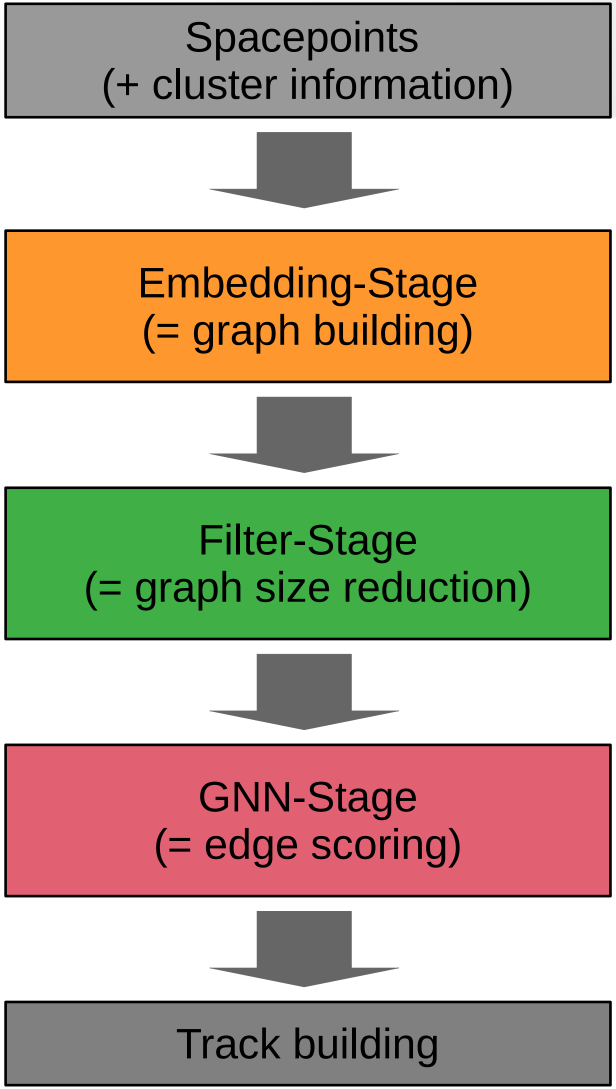
28.09.2022

Benjamin Huth
Universität Regensburg

Overview

- Introduction to Exa.TrkX pipeline
- Tutorial
  - Generate training data
  - Configure & run training
  - Monitor & tune training performance
  - Convert models
  - Run inference in ACTS

The Exa.TrkX pipeline



- Multi-stage machine-learning pipeline for **track finding**
  - Event as a graph (nodes = hits, edges = potential track segments)
  - Use GNNs to find edges that correspond to track segments
- See talk by Xiangyang yesterday

Prerequisites

```
ACTS_BUILD_EXAMPLES_EXATRKX:BOOL=ON
ACTS_BUILD_PLUGIN_EXATRKX:BOOL=ON
ACTS_EXATRKX_ENABLE_ONNX:BOOL=OFF
ACTS_EXATRKX_ENABLE_TORCH:BOOL=ON
```

- Build acts with torchscript backend
- Docker images for training & inference under github.com/acts-project/machines
  - Take a look at the Dockerfile to for precise install instructions
- Training:
  - CUDA, torch + family, pytorch-lightning, *traintrack**, faiss, frnn (install from github)
  - GPU with lots of memory (~30 GB for ODD event with 200 pileup)
- ACTS inference:
  - CUDA, libtorch, torch-scatter (needs to be compiled locally)
  - Not so much GPU memory (~14 GB for ODD event with 200 pileup)

Training data generation

- Slightly modified Exa.TrkX code works with ACTS CSV output
- Required writers:
  - `CsvTrackingGeometryWriter`
  - `CsvParticleWriter`
  - `CsvSimHitWriter`
  - `CsvMeasurementWriter` *(optional)*
- Fixed directory structure
- Can be trained using truth hits or measurement data
  - Switch in the `processing.yaml`

```
data
├── detectors.csv
└── train_all
    ├── event000000000-cells.csv
    ├── event000000000-measurements.csv
    ├── event000000000-measurement-simhit-map.csv
    ├── event000000000-particles.csv
    ├── event000000000-truth.csv
    ├── event000000001-cells.csv
    ├── event000000001-measurements.csv
    ├── event000000001-measurement-simhit-map.csv
    ├── event000000001-particles.csv
    ├── event000000001-truth.csv
```

Configure the training

- Training steered by *traintrack* (by Daniel Murnane)
  - Project configuration at `./configs/project_config.yaml`
  - Some other options possible (for schedulers etc.)

```yaml
# Location of libraries
libraries:
    model_library: LightningModules
    artifact_library: tmp

# Which logger to use - options are Weights & Biases [wandb], TensorBoard [tb], or [None]
logger: wandb
```

Configure the training

```
stage_list:
    - {set: Processing, name: TrackMLFeatureStore, config: processing.yaml}
    - {set: Embedding, name: LayerlessEmbedding, config: embedding.yaml}
    - {set: Filter, name: VanillaFilter, config: filter.yaml, resume_id: 1suumxob}
#     - {set: GNN, name: InteractionGNN, config: gnn.yaml}
```

- State list defines
  - „set“ (basically subfolder)
  - Python class (in `<Set>/Models/`)
  - Configuration file (in `<Set>`)
  - resume_id
  - Allows to overwrite hyperparameters

```
LightningModules
├── Embedding
│   ├── Models
│   │   ├── inference.py
│   │   └── layerless_embedding.py
│   ├── embedding_base.py
│   └── embedding.yaml
├── Filter
│   ├── Models
│   │   ├── inference.py
│   │   └── vanilla_filter.py
│   ├── filter_base.py
│   └── filter.yaml
├── GNN
│   ├── Models
│   │   ├── inference.py
│   │   └── interaction_gnn.py
│   ├── gnn_base.py
│   └── gnn.yaml
└── Processing
    ├── Models
    │   └── feature_construction.py
    ├── utils
    ├── feature_store_base.py
    └── processing.yaml
```

Configure the training

```yaml
# Input/output configuration
input_dir: ${PROCESSING_OUTPUT}
output_dir: ${EMBEDDING_OUTPUT}
project: ${PROJECT_NAME}-embedding
overwrite: True

# Dataset parameters
pt_signal_cut: 0.0
pt_background_cut: 0.0
train_split: [[950,25,25]] # [train, val, test]
true_edges: modulewise_true_edges
noise: False

# Model parameters
spatial_channels: 3
cell_channels: 0
emb_hidden: 1024
nb_layer: 4
emb_dim: 8
weight: 2
activation: Tanh
randomisation: 4
points_per_batch: 130000
r_train: 0.2
r_val: 0.2
r_test: 0.2
knn: 50
warmup: 4
margin: 0.1
lr: 0.0003
factor: 0.58
patience: 19
regime: [[rp, hnm, norm]]
max_epochs: 20

# Postprocessing
callbacks: [[EmbeddingTelemetry, EmbeddingBuilder]]
```

- Lots of parameters
  - Does evaluate environment variables
  - `project` used e.g., by wandb
  - callbacks generate output
  - [[…]] necessary for list

Run the training

- Set environment variables like `$EXATRKX_DATA`
- Run: `$ traintrack <pipeline-config-file>`
- Can take very long dependent on data/event size
- If embedding/filter stages do not perform well enough, GNN stage can fail due to memory requirements
- **Note:** Training example of workshop does NOT aime for good performance, should just run in a few minutes

Resume from checkpoint

- Runs aborted due to connection errors etc. can be resumed:
  - Find out run id:

```
tmp/workshop-demo-embedding/us5xg97y/checkpoints
```

  - Change `pipeline.yaml` accordingly

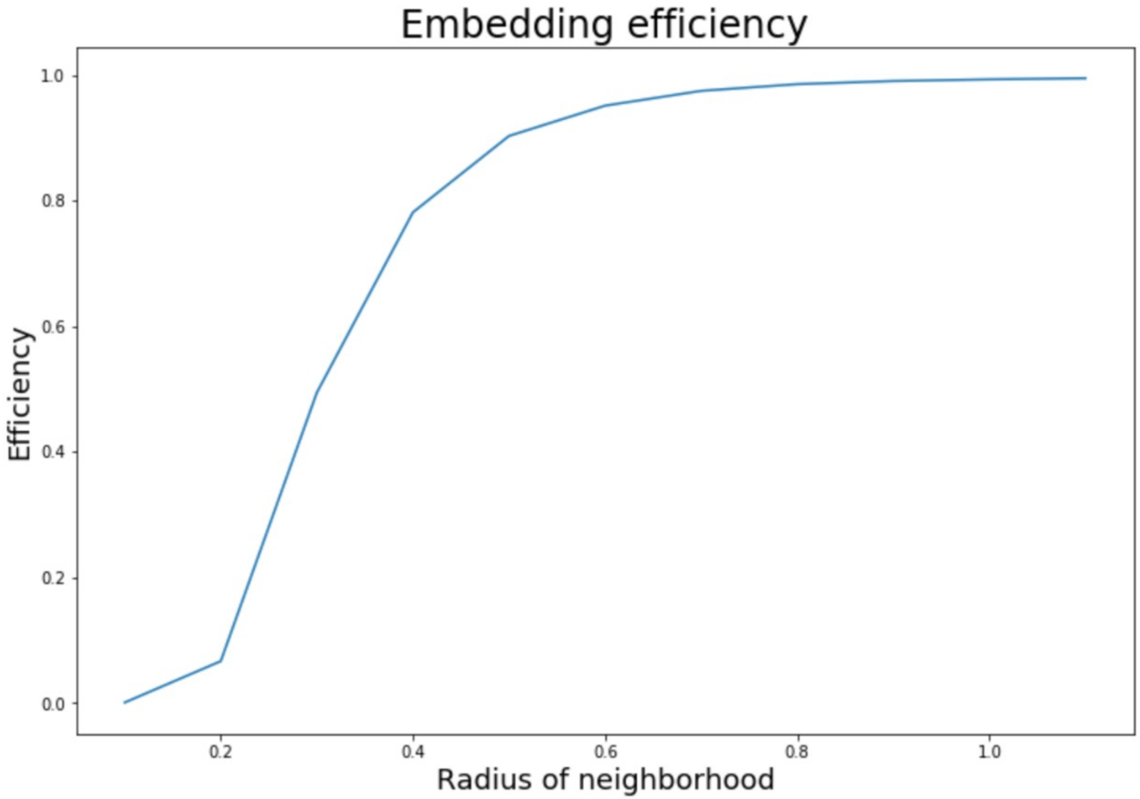
```
stage_list:
#    - {set: Processing, name: TrackMLFeatureStore, config: processing.yaml}
    - {set: Embedding, name: LayerlessEmbedding, config: embedding.yaml, resume_id: us5xg97y}
    - {set: Filter, name: VanillaFilter, config: filter.yaml}
    - {set: GNN, name: InteractionGNN, config: gnn.yaml}
```

Monitor metrics

- wandb logger allows real-time monitoring,etc.

Tune training

- Important HPs to controll efficiency/purity:
  - Embedding: radius
  - Filter: filter_cut
  - GNN: edge_cut
- Callbacks produce analysis graphs as *.pdf
  - Can also be done e.g. in a jupyter notebook (see here)
- General HP tuning possible with wandb



Convert to torchscript

- Straight forward:
  - load model from checkpoint (see here)
    - Look for `last.ckpt` in the `tmp/<project>/<run-id>` directory
  - Call `model.eval()` to leave training mode
  - Convert to torchscript (see here)
    - Prepare example input if use tracing mode:

```
n_nodes = 10
x = torch.rand(n_nodes,3)
edge_index = torch.randint(0,n_nodes,(2,10))
```

Run inference

- Setup Trackfinding algorithm:
  - Make spacepoints for whole detector
    - Needs geometry selection of whole detector
  - Add ExaTrkX algorithm
    - See also [here]
    - Ensure hyperparameter match the trained model
- CPU only not really possible now with the ACTS implementation

```
exaTrkXConfig = {
    "modelDir": str(modelDir),
    "spacepointFeatures": 3,
    "embeddingDim": 8,
    "rVal": 0.05,
    "knnVal": 500,
    "n_chunks": 12,
    "filterCut": 0.01,
    "edgeCut": 0.5
}
```

Evaluate performance

- Use `TrackFinderPerformanceWriter`
- Two `TTrees`
  - `track_finder_particles`: particle based metrics

event_id	particle_id	particle_type	vx	vy	vz	vt	px	py	pz	m	q	nhits	ntracks	ntracks_majority
0	4503599677702144	-211	0.00e+00	0.00e+00	0.00e+00	0.00e+00	1.32	0.07	-24.55	1.40e-01	-1.0	5	1	1
0	4503599694479360	211	0.00e+00	0.00e+00	0.00e+00	0.00e+00	-0.49	0.21	3.87	1.40e-01	1.0	11	1	1
0	4503599711256576	-211	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.56	0.37	2.79	1.40e-01	-1.0	13	1	1

  - `track_finder_tracks`: track based metrics

event_id	track_id	size	nparticles	particle_id	particle_nhits_total	particle_nhits_on_track
0	40	8	2	[680104020496351232, 765996792901795840]	[5, 3]	[5, 3]
0	43	8	2	[689050744396972032, 855683929217171456]	[4, 4]	[4, 4]
0	134	10	2	[833182425969328128, 770149623439294464]	[10, 1]	[9, 1]

Evaluate performance

- Then do the analysis like you want
  - I have a jupyter notebook (see provided files)

```
[20]: fig, ax = plt.subplots(1,3, figsize=(18,5))
fig.suptitle("Efficiency")

ax[0].hist(particles_df.efficiencies)
ax[0].set_xlabel("efficiency")
#ax[0].scatter(particles_df.eta, particles_df.reconstructed, alpha=0.01)
ax[1].scatter(particles_df.eta, particles_df.efficiencies, alpha=0.005)
ax[1].set_xlabel("eta")
ax[1].set_ylabel("efficiency")
ax[2].scatter(particles_df.pT, particles_df.efficiencies, alpha=0.005)
ax[2].set_xlabel("pT")
ax[2].set_ylabel("efficiency")
_ = ax[1].set_ylabel("efficiency")
```

How can I do this at home?

- There is some code attached to the indico session
  - Some modifications compared to upstream training code of Exa.TrkX to support
    - mainly in the Preprocessing to support latest Writers
- I will add a „How to“ soon to the ACTS documentation
- As well as a proper repository/fork with analysis scripts
- There will be soon a training CI in ACTS that could also be used as reference
- There are also tutorial-notebooks by Exa.TrkX (see here)
- If you encounter any problems:
  - E-mail: benjamin.huth@ur.de
  - Contact me on ACTS Mattermost channel

Developement Outlook

- Cluster information cannot be used yet
  - Should come soon
- More fine-grained interfaces for graph-building and edge-labeling (see talk by Xiangyang yesterday)
  - To allow for composable algorithms