Co-funded by the Horizon 2020 Framework Programme of the European Union

Grant agreement No. 671555

ExCAPE
Exascale Compound Activity Prediction Engine

Future and Emerging Technologies (FET)

Call: H2020-FETHPC-2014
Topic: FETHPC-1-2014
Type of action: RIA

Deliverable D2.21

Report: Final Simulation and Scalability Report

Scalability of ExCAPE pipelines

Due date of deliverable: 31.08.2018
Actual submission date: 31.08.2018

Start date of Project: 1.9.2015 Duration: 36 months

Responsible Consortium Partner: Intel
Contributing Consortium Partners: IT4I, IMEC, Intel
Name of author(s) and contributor(s): Vojtěch Cima (IT4I), Tom Vander Aa (IMEC), Yves Vandriessche (INTEL)
Internal Reviewer(s): Jiří Dvorský, Kateřina Janurová (IT4I), Imen Chakroun (IMEC)
Revision: V2

Project co-funded by the European Commission within Horizon 2020 Framework Programme (2014-2020)		
Dissemination Level		
PU	Public	PU
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Document revision tracking

This page is used to follow the deliverable production from its first version until it has been reviewed by the assessment team. Please give details in the table below about successive releases.

Release number	Date	Reason of this release and/or validation	Dissemination of this release (task level, WP/ST level, Project Office Manager, Industrial Steering Committee, etc)
0	10.7.2018	First version	CO
1	24.8.2018	Internal review	CO
2	31.8.2018	Final version	PU

Glossary

No glossary is given.

Link to Tasks

Task number	Work from task carried out	Deviations from task technical content, with motivation and summary of impact
2.4.1	Provide a highly scalable, reference implementation of selected standard machine-learning algorithms	None
2.4.2	Provide a highly scalable, reference implementation of new machine learning algorithms	None
2.4.3	Optimization of machine learning programs to benefit from specific HPC resources	None
2.5.1	Scalability benchmarking	None

Contents

Executive Summary

This report presents the machine learning pipelines used for hyperparameter search for ExCAPE models.

We performed large scale experiments using five machine learning methods, two multi-task methods, three single task methods.

We used in total 238 TB of disk space, most of it for the multi-task methods, and consumed 4.7 million core hours.

Dissemination and Exploitation

All the modelling pipelines have been used to create predictive models that have been extensively tested in WP3. Given that these pipelines are easily parametrizable, they may be exploited for further experiments beyond the scope of this project.

1 Introduction

This deliverable overviews the machine learning based data processing pipelines used to build and validate ExCAPE models on large-scale distributed infrastructure provided by IT4I.

This deliverable focuses on compute performance and HPC scalability. Please note that we do *not* report on any of the machine learning aspects in this deliverable. We do not go in any way into the quality of the models and their use in chemogenomics. Machine learning results are discussed in deliverable D3.18 of WP3 [4].

This document is structured as follows. Section 2 describes multi-task learner pipelines (SMURFF, ExNET), Section 3 describes single-task learner pipelines (gradient boosting, support vector machine, random forest). Section 4 forms the conclusions.

2 Multi-task Learners

Two multi-task methods have been used in ExCAPE: deep learning and matrix factorization. In the first subsection (Section 2.1) we explain the matrix factorization experiments and their scalability done with SMURFF, in the second subsection (Section 2.2) we do the same for deep learning with ExNET.

2.1 SMURFF

In this section we describe the SMURFF scalability experiments that were performed on the ExCAPE dataset (v5) using the HyperLoom work-flow framework. In the following subsection we explain the cross-fold validation setup used and its corresponding HyperLoom pipeline (Section 2.1), the experimental setup on the IT4I infrastructure used to perform SMURFF runs (Section 2.1), scalability metrics and results we collected from those runs (Section 2.1), and finally lessons learned and conclusion (Section 2.1).

SMURFF pipeline

We performed a full cross-fold validation to find and verify the best hyperparameters. As already mentioned, the machine learning aspects of this process are described in D3.18 of WP3 [4].

For example, for the inner fold validation, we have explored 37 hyperparameter combinations for 6 folds, resulting in 222 runs of SMURFF.

The HyperLoom plan is depicted in Figure 1. The different HyperLoom task types are:

1. Loading the side info matrix from disk.
2. Loading test and train data from disk.
3. Remove empty rows form the matrices.
4. Centering the bio-activity values around the global mean.
5. Running SMURFF with the centered data on the selected hyperparameters.

The plan shown in the Figure 1 has been coded in Python use the HyperLoom and SMURFF Python API, allowing for easy integration of the two software packages.

We have made a similar HyperLoom pipeline for the outer folds.

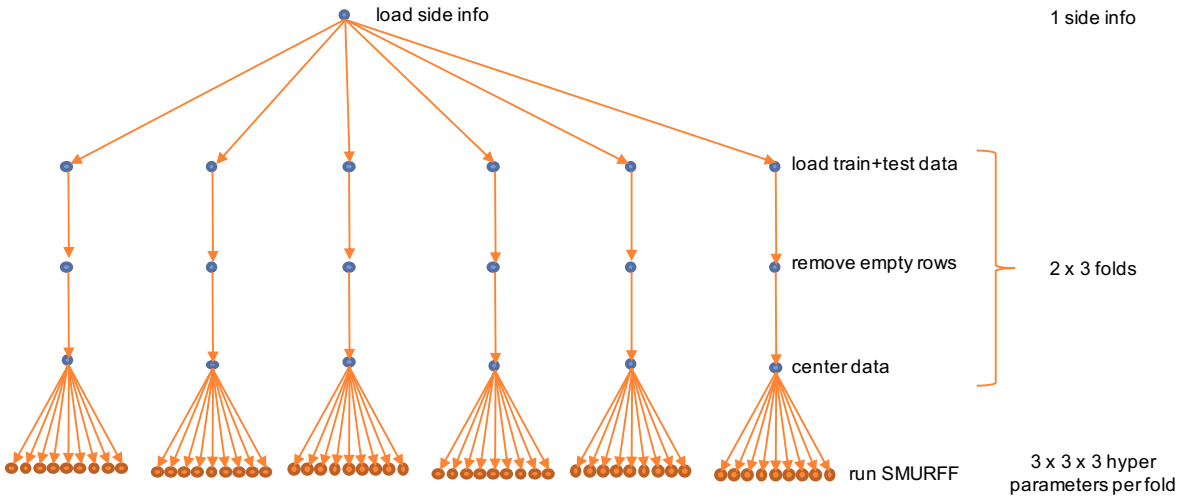

Figure 1: The SMURFF+HyperLoom pipeline for the inner folds.

SMURFF on IT4I

As described in deliverable D2.19 [1], the Salomon cluster on IT4I has built-in support to run HyperLoom pipelines. The ExCAPE public data is also available on Salomon. This made it easy to run the SMURFF + HyperLoom pipeline. However, jobs on Salomon are limit to a total time of 48 hours, too little for the complete pipeline to finish. Luckily SMURFF has built in support for check-pointing (as described in D2.14 [2]), allowing us to continue the runs in a newly submitted job from the checkpoint.

SMURFF Scalability

During the inner fold, outer fold and full data runs we collected various statistics to measure the robustness, performance and scalability of SMURFF and of the SMURFF + HyperLoom pipeline. We discuss these statistics in this section.

Conjugate Gradient Solver We selected the Macau algorithm [3] for all the SMURFF runs, and started with a run learning from the side info (the ECFP chemical compound descriptors) using the CG solver method. This is one of the two methods in Macau to learn from side info. The CG solver method works very well for many features per compound.

After several experimental runs however, we realized this method would be too slow. Indeed, a full hyperparameter exploration for the inner folds using the CG solver would consume more than 1M core hours, and one single run would take more than 8 days.

Direct Method Luckily SMURFF also supports a direct inversion based method to incorporate side information. This method requires much more memory, and is unfeasible for large side info matrices, but thanks to the 128GB of RAM on each Salomon node, we could use this method for our experiments.

Statistics In total we submitted three HyperLoom plans (inner, outer, and full) with 703 HyperLoom tasks in total, consuming 180,000 core hours of compute, running on up 100 nodes in parallel. Out of the 703 tasks, 33 are data preparation, 335 are training tasks, and 335 are prediction tasks on the test set.

Table 3: Statistics of SMURFF runs for hyperparameter exploration

	inner	outer	full	total
no. of samples (including burnin)	2,000	2,000	2,000	
no. of folds	6	3	1	
no. of hyperparameters	37	37	2	
data prep. tasks	19	10	4	33
train tasks	222	111	2	335
prediction tasks	222	111	2	335
core hours (×1000)	118	59	2	178
disk space per model (GB)	580	850	1,700	
disk space total (TB)	126	92	3	221

We produced 335 models with SMURFF. Each model consumes between 508 and 1,700 GB in disk space, for a total disk space used of 221 TB. These statistics are summarized in Table 3.

SMURFF Lessons Learned

Doing these large scale SMURFF runs learned us these lessons:

- Using the CG solver method on the ExCAPE dataset is intractable.
- HyperLoom integration with SMURFF is fast and efficient thanks to the Python API of both packages.
- The SMURFF built-in check-pointing works very well and helped us substantially to restart and finish all the runs.
- Disk-space is more limiting than compute hours. We need a more space-efficient way to store the SMURFF models, not only to conserve disk space of Salomon, but also for archiving purposes and to be able to transfer the models to the ExCAPE partners.

2.2 ExNET

This modelling pipeline performs a hyperparameter sweep for ExNET multi-task learning models on the ExCAPE dataset (v5). We report on two different runs, one using the sparse ECFP feature descriptors and another using the dense Chem2Vec.

ExNET pipeline with ECFP descriptors Table 4 shows the hyperparameter grid for the ExNET pipeline used for runs with ECFP descriptors feature matrices. Table 5 presents pipeline execution details.
Features matrix: `data_v5_exnet/data/side_info/ECFP6_counts_var005.mtx`
Data directory: `data_release_v5/exnet/data`

Table 4: ExNET hyperparameter grid (ECFP)

parameter	value
learning_rate	0.01, 0.1
momentum	0.0, 0.4
network_layout	(2048, 2048), (1024, 1024, 1024), (2048, 2048, 2048), (4096, 4096, 4096)
dropout_keep_rate	0.5
epochs	200
input_dropout_keep_rate	0.8, 1.0
undersampling	none
scaling	none

Table 5: Statistics of ExNET runs for hyperparameter exploration (ECFP)

	inner	outer	full	total
no. of folds	6	3	1	
no. of hyperparameters	32	32	1	
train tasks	192	96	1	289
prediction tasks	192	96	1	289
core hours (×1,000)				450
disk space total (TB)	9.5	6.2		15.7

Table 6: XGBoost hyperparameter grid (ECFP)

parameter	value
objective	"binary:logistic"
booster	"gbtree"
learning_rate	0.05
scale_pos_weight	1, 5, 10
n_estimators	50, 100, 200
max_depth	5, 10
undersampling	none
scaling	none

ExNET pipeline with C2V descriptors ExNET runs with C2V descriptors have been reported in Deliverable D3.18. These runs have used approximately 150k core hours of compute time and resulting models occupy 700 GB of disk space in total.
Features matrix:

- `data_v5_exnet/data/side_info/chem2vec40prod_std.mtx`
- `data_v5_exnet/data/side_info/chem2vec40sum_std.mtx`
- `data_v5_exnet/data/side_info/chem2vec_std.mtx`

Data directory: `data_release_v5/exnet/data`

3 Single-task Learners

Three single-task methods were tested on a large scale in ExCAPE: Gradient Boosting, Support Vector Machines, and Random Forest. These three methods will be described in the next subsections.

3.1 Gradient Boosting

This modelling pipeline performs hyperparameter sweep on ExCAPE dataset (v5) using XGBoost[1] (single-task learner). The experiments were performed using ECFP and C2V descriptors.

XGBoost pipeline with ECFP descriptors Table 6 shows the hyperparameter grid for the XGBoost pipeline used for runs with ECFP descriptors. Table 7 presents pipeline execution details.
Feature matrix: `data_v5_exnet/data/side_info/ECFP6_counts_var005.mtx`
Data directory: `data_release_v5/exnet/data`

XGBoost pipeline with C2V descriptors Table 8 shows the hyperparameter grid for the XGBoost pipeline used for runs with C2V descriptors. Table 9 presents pipeline execution details.
Feature matrix: `data_release_v5/exnet/data/side_info/chem2vec.mtx`
Data directory: `data_release_v5/exnet/data`

[1] https://xgboost.readthedocs.io/en/latest/index.html

Table 7: Statistics of XGBoost runs for hyperparameter exploration (ECFP)

	inner	outer	full	total
no. of folds	6	3	1	
no. of hyperparameters	18	18	1	
no. of targets	526	526	526	
train tasks	56,808	28,404	526	85,738
prediction tasks	56,808	28,404	526	85,738
core hours (×1,000)				3
disk space total (GB)				399

Table 8: XGBoost hyperparameter grid (C2V)

parameter	value
objective	"binary:logistic"
booster	"gbtree"
learning_rate	0.05
scale_pos_weight	1, 5, 10
n_estimators	50, 100, 200
max_depth	5, 10
undersampling	none
scaling	none

Table 9: Statistics of XGBoost runs for hyperparameter exploration (C2V)

	inner	outer	full	total
no. of folds	6	3	1	
no. of hyperparameters	18	18	1	
no. of targets	526	526	526	
train tasks	56,808	28,404	526	85,738
prediction tasks	56,808	28,404	526	85,738
core hours (×1,000)				5
disk space total (GB)				385

Table 10: SVM hyperparameter grid (ECFP)

parameter	value
kernel	"linear"
C	300, 100, 30, 10, 1, 0.1, 0.05, 0.01, 0.001
penalty_loss_dual	("l2", "squared_hinge", True), ("l2", "squared_hinge", False), ("l2", "hinge", True), ("l1", "squared_hinge", False)
undersampling	none
scaling	MaxAbsScaler

Table 11: Statistics of SVM runs for hyperparameter exploration (ECFP)

	inner	outer	full	total
no. of folds	6	3	1	
no. of hyperparameters	36	36	1	
no. of targets	526	526	526	
train tasks	113,616	56,808	526	170,950
prediction tasks	113,616	56,808	526	170,950
core hours (×1,000)				3
disk space total (GB)				407

3.2 Support Vector Machine

This modelling pipeline performs hyperparameter sweep on ExCAPE dataset (v5) using SVM - LinearSVC [2] (single-task learner). The experiments were performed using ECFP and C2V descriptors.

SVM pipeline with ECFP descriptors Table 10 shows the hyperparameter grid for the SVM pipeline used for runs with ECFP descriptors. Table 11 presents pipeline execution details.
Feature matrix: `data_v5_exnet/data/side_info/ECFP6_counts_var005.mtx`
Data directory: `data_release_v5/exnet/data`

SVM pipeline with C2V descriptors Table 12 shows the hyperparameter grid for the SVM pipeline used for runs with C2V descriptors. Table 13 presents pipeline execution details.
Feature matrix: `data_release_v5/exnet/data/side_info/chem2vec_std.mtx`
Data directory: `data_release_v5/exnet/data`

[2] http://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html

Table 12: SVM hyperparameter grid (C2V)

parameter	value
kernel	"linear"
C	300, 100, 30, 10, 1, 0.1, 0.05, 0.01, 0.001
penalty_loss_dual	("l2", "squared_hinge", True), ("l2", "squared_hinge", False), ("l2", "hinge", True), ("l1", "squared_hinge", False)
undersampling	none
scaling	none

Table 13: Statistics of SVM runs for hyperparameter exploration (C2V)

	inner	outer	full	total
no. of folds	6	3	1	
no. of hyperparameters	36	36	1	
no. of targets	526	526	526	
train tasks	113,616	56,808	526	170,950
prediction tasks	113,616	56,808	526	170,950
core hours (×1,000)				3
disk space total (GB)				407

3.3 Random Forest

This modelling pipeline performs hyperparameter sweep on ExCAPE dataset (v5) using random forest classifier[3] (single-task learner). The experiments were performed using ECFP and C2V descriptors.

RF pipeline with ECFP descriptors Table 14 shows the hyperparameter grid for the RF pipeline used for runs with ECFP descriptors. Table 15 presents pipeline execution details.
Feature matrix: `data_v5_exnet/data/side_info/ECFP6_counts_var005.mtx`
Data directory: `data_release_v5/exnet/data`

[3]http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

Table 14: RF hyperparameter grid (ECFP)

parameter	value
n_estimators	128, 256, 512
max_features	"auto"
class_weight	"balanced"
undersampling	none
scaling	none

Table 15: Statistics of RF runs for hyperparameter exploration (ECFP)

	inner	outer	full	total
no. of folds	6	3	1	
no. of hyperparameters	3	3	1	
no. of targets	526	526	526	
train tasks	3,156	1,578	526	5,260
prediction tasks	3,156	1,578	526	5,260
core hours (×1,000)				10
disk space total (GB)				585

Table 16: RF hyperparameter grid (C2V)

parameter	value
n_estimators	128, 256, 512
max_features	"auto"
class_weight	"balanced"
undersampling	none
scaling	none

RF pipeline with C2V descriptors Table 16 shows the hyperparameter grid for the RF pipeline used for runs with C2V descriptors. Table 17 presents pipeline execution details.
Feature matrix: `data_release_v5/exnet/data/side_info/chem2vec.mtx`
Data directory: `data_release_v5/exnet/data`

4 Conclusion

At the end of the ExCAPE project, we performed large scale experiments using five machine learning methods, two multi-task and three single task methods.

We used in total 238 TB of disk space, most of it for the multi-task methods, and consumed approximately 4.7M core hours in total including all ExCAPE experiments

Table 17: Statistics of RF runs for hyperparameter exploration (C2V)

	inner	outer	full	total
no. of folds	6	3	1	
no. of hyperparameters	3	3	1	
no. of targets	526	526	526	
train tasks	3,156	1,578	526	5,260
prediction tasks	3,156	1,578	526	5,260
core hours (×1,000)				5
disk space total (GB)				164

executed at IT4I supercomputing facility.

References

[1] Stanislav Böhm (IT4I) and Vojtěch Cima (IT4I). ExCAPE deliverable D2.19: Other: Framework 6. Technical report, 2018.

[2] Stanislav Böhm (IT4I), Vojtěch Cima (IT4I), Tom Vander Aa (IMEC), and Yves Vandriessche (Intel). ExCAPE deliverable D2.14: Other: Framework 4. Technical report, 2018.

[3] Jaak Simm, Adam Arany, Pooya Zakeri, Tom Haber, Jörg K. Wegner, Vladimir Chupakhin, Hugo Ceulemans, and Yves Moreau. Macau: Scalable bayesian multi-relational factorization with side information using MCMC, 2015.

[4] Noé Sturm, Hongming Chen, Vladimir Chupakhin, and Nina Jeliazkova. ExCAPE deliverable D3.18: Benchmark report 5. Technical report, 2018.