UiO : University of Oslo

Decision making in underground construction

Variations of ML – models + some hard learnt lessons

Tom F. Hansen

NGI DIGITAL

Image from Bane NOR

Agenda

- Why I did this
- Papers
- Some “non-science” experiences
- Software based research is still research
- The dataset part – variations of ML models
- The simulation part – Reinforcement learning

UiO : University of Oslo

Papers

- : Datascience
- : Supervised learning
- : Unsupervised learning
- : Reinforcement learning
- : Explainable AI

Paper I: Building and analysing a labelled measure while drilling dataset from 15 hard rock tunnels in norway.
T.F. Hansen, Z. Liu, J. Torresen
In review in journal "Tunneling and Underground Space Technology", 2024.
Preprint at SSRN:http://dx.doi.org/10.2139/ssrn.4729646

Paper II: Improving face decisions in tunnelling by machine learningbased MWD analysis.
T. F. Hansen, G. H. Erharter, T. Marcher, Z. Liu, and J. Tørresen
Geomechanics and Tunnelling, vol. 15, no. 2, pp. 222–231, 2022.
DOI:10.1002/geot.202100070

Paper III: Predicting rock type from mwd tunnel data using a reproducible ml-modelling process.
T.F. Hansen, Z. Liu, J. Tørresen
"Tunneling and Underground Space Technology", 2024.
DOI: https://doi-org./10.1016/j.tust.2024.105843

Paper IV: A comparative study on machine learning approaches for rock mass classification using drilling data.
T.F. Hansen, G.H. Erharter, Z. Liu, J. Torresen
In review in journal "Applied computing and geosciences", 2024. Preprint arXiv:http://arxiv.org/abs/2403.10404.

Paper V: Can we trust the machine learning based geotechnical model?
T.F. Hansen
Proceedings of the conference 5th ICITG, 2024, Colorado School of Mines, USA. Public proceedings 30.06.2024.

Paper VI: Unsupervised machine learning for data-driven classification of rock mass using drilling data.
T.F. Hansen, A. Aarset
In review in journal "Rock mechanics and rock engineering". Preprint arXiv:http://arxiv.org/abs/2403.10404.

Paper VII: Reinforcement learning based process optimization and strategy development in conventional tunnelling.
G.H. Erharter, T.F. Hansen, Z. Liu, T. Marcher
Automation in Construction, volume 127, 2021.
DOI:10.1016/j.autcon.2021.103701

Paper VIII: TunnRL-CC: A computational framework for smart TBM cutter changing.
T.F. Hansen, G. Erharter, T. Marcher
"Automation in construction", volume 165, 2024.
DOI: 10.1016/j.autcon.2024.105505.

UiO : University of Oslo

Papers during Phd

Papers written during the PhD project, not included in the thesis
Paper: International distribution and development of rock mass classification - a review
G. Erharter, N. Bar, T.F. Hansen, S. Jain, T. Marcher
Submit for review to the journal "Rock mechanics and rock engineering"..

Paper: A 2023 perspective on Rock Mass Classification Systems
G. Erharter, T.F. Hansen, S. Qi, N. Bar, T. Marcher
Conference: 15th ISRM Congress 2023 & 72nd Geomechanics Colloquium, Salzburg, Austria

Paper: Towards optimized TBM cutter changing policies with reinforcement learning
G. Erharter, T.F. Hansen
Geomechanics and Tunnelling, vol. 15, no. 2, pp. 665-670, 2022.
DOI:https://doi.org/10.1002/geot.202200032

Paper: Analysis of water ingress, grouting effort, and pore pressure reduction caused by hard rock tunnels in the Oslo region
J. Langford, K. Holmøy, T.F. Hansen, K.G. Holter, E. Stein
Tunnelling and Underground Space Technology incorporating Trenchless Technology Research, vol. 130, 2022.
DOI:https://doi.org/10.1016/j.tust.2022.104762

Paper: Introducing Reinforcement Learning to Tunneling
G. Erharter, T.F. Hansen, Z. Liu, T. Marcher
Conference: International conference on Computational methods and information models in tunnelling, Bochum, Germany, 2022.

Paper: Norwegian tunnel excavation: Increasing digitalisation in all operations
J.K.Y. Chiu, T.F. Hansen, T. Wetlesen
Geomechanics and Tunnelling, vol. 15, no. 2, pp. 182-189, 2022.
DOI:https://doi.org/10.1002/geot.202100072

Github repositories with code supporting the papers
- https://github.com/tfha/MWD-dataset
- https://github.com/tfha/ML-MWD-prediction-tabular
- https://github.com/tfha/ML-MWD-prediction-rocktype
- https://github.com/tfha/ML-MWD-prediction-images
- https://github.com/tfha/ML-MWD-clusterings
- https://github.com/TunnRL/TunnRL_TBM_maintenance

Be nice with the laptop

In many ways, the field of machine learning can be said to be just as close to HPC computing (with its focus on hardware and heavy computation) as classic software development. Like HPC workloads, machine learning workloads often will benefit from faster execution and quicker experimentation when running on an HPC machine.

These features make an HPC a better choice than your local laptop for ML training.

- A remote HPC machine typically have more cores and a faster CPU than your laptop. More cores let you run code in parallelization faster. If your progress bar reach 1% done after 1 hour, you know what to do.
- Efficient cooling system. Massive ML training is not good for your laptop. Listen to the fan and feel the temperature 🔥. Have some empathy with your computer.
- More memory. Memory is important, especially in training neural networks with millions of parameters that need to be stored. Too little memory will crash your runs and freeze your computer.
- An HPC machine might have one or several strong GPU's with lots of memory. These are the go-to machines for computer vision tasks and NLP. For images larger than 10x10 pixels, this will take forever without a GPU.
- Training on an HPC also keeps your laptop ready to do other work (and not to break down 😉.

Docker is your friend

The biggest problem, though, is to successfully get the dependencies, the python version, and the tools you have installed ++, so you **actually** can run your script. If you have a simple script and only use a Numpy dependency, this might work, but ML-training scripts are not like that. To take advantage of a cluster for machine learning training, you'll need to ensure your development environment is portable and training is reproducible on an HPC.

The solution to your problems and to run ML training in an efficient and less nerve-breaking way, you should containerize your code and then run it on the remote. Docker is your friend.

Code academy description – HPC + docker for ML

https://ngiwiki.slite.com/app/docs/zM8sK924BSt990

https://ngiwiki.slite.com/app/docs/lI_xi7DmodNoIB

UiO : University of Oslo

My coding journey

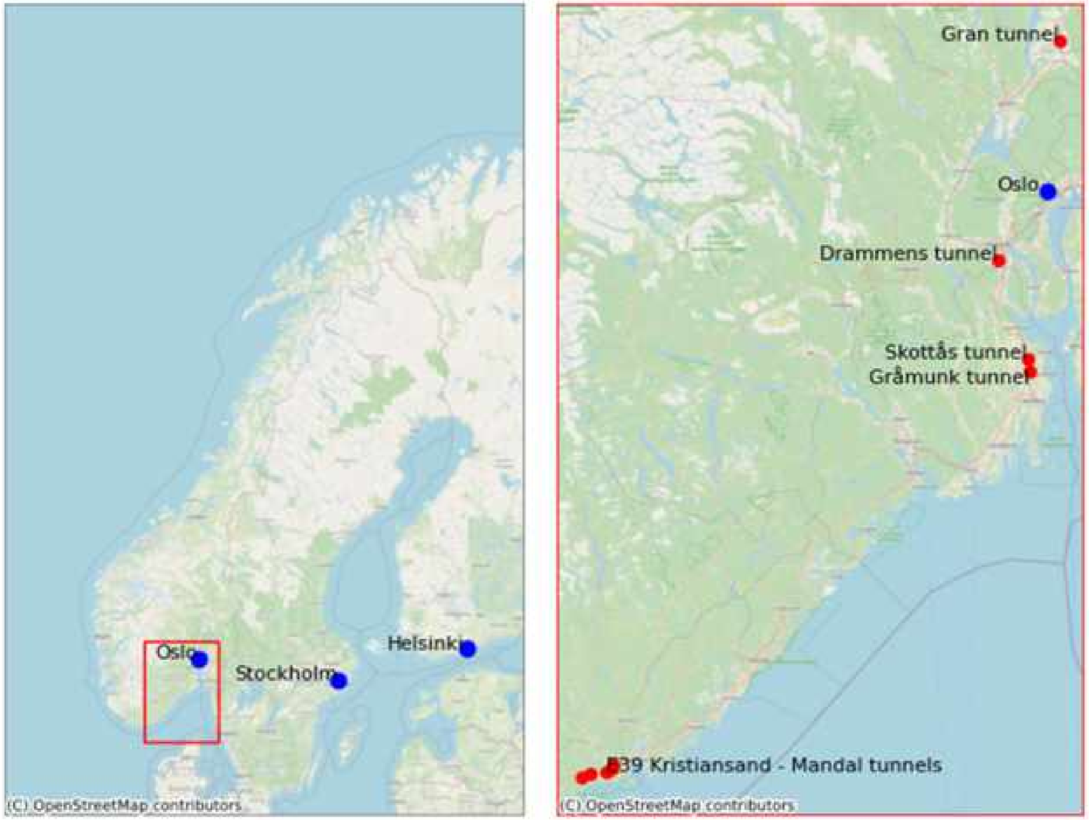
```
mask_label: [
    "B",
    "D",
        ]

combine_labels:
    A: A
    B: B
    C: C
    D: D
    E1: E
    E2: E
```

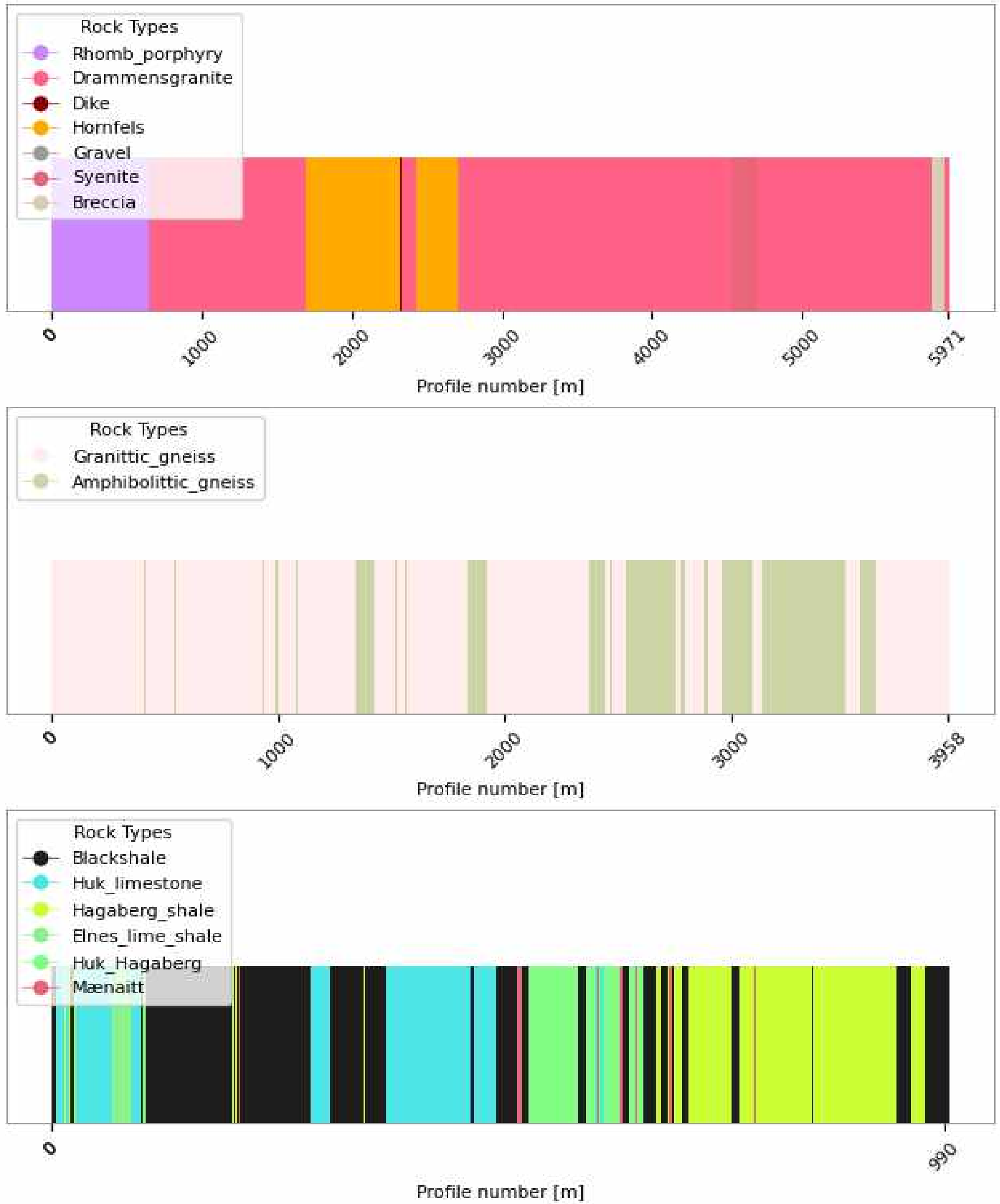
```python
if mask_labels is not None:
    mask = df[label].isin(mask_labels)
    df = df.loc[~mask, :]

if combine_labels is not None:
    df[label] = df[label].map(combine_labels)
```

```python
def transform_label(
    data: pd.DataFrame,
    transform: str = None,
    features: List[str] = [],
    label: str = "Q_class",
    visualize=True,
    q_base=False,
):
    dataset = data.copy().sample(
        frac=1, random_state=42
    )  # resamples to get meaningful dataset
    dataset = dataset.reset_index(drop=True)
    labels = dataset[label]
    if not transform:
        if visualize:
            print(labels.value_counts())
        return dataset[features], labels
    else:
        labels = dataset.q.apply(calculate_class)  # full split in classes
        if transform == "A_B_C_D_E1_E2":
            labels = labels
        elif transform == "AB_CD_E":
            if q_base:
                labels = dataset.q_base.apply(calculate_class)
            labels = labels.str.replace("A", "AB")
            labels = labels.str.replace("B", "AB")
            labels = labels.str.replace("C", "CD")
            labels = labels.str.replace("D", "CD")
            labels = labels.str.replace("E1", "E")
            labels = labels.str.replace("E2", "E")
            labels = labels.str.replace("AAB", "AB")
            labels = labels.str.replace("CCD", "CD")
        elif transform == "A_B_C_D_E":
            labels = labels.str.replace("E1", "E")
            labels = labels.str.replace("E2", "E")
        elif transform == "ABCD_E":
            labels = labels.str.replace("A", "ABCD")
            labels[labels == "B"] = labels[labels == "B"].str.replace("B", "ABCD")
            labels[labels == "C"] = labels[labels == "C"].str.replace("C", "ABCD")
            labels[labels == "D"] = labels[labels == "D"].str.replace("D", "ABCD")
            labels[labels == "E1"] = labels[labels == "E1"].str.replace("E1", "ABCD")
        elif transform == "AB_CDE":
            labels[labels == "A"] = labels[labels == "A"].str.replace("A", "AB")
            labels[labels == "B"] = labels[labels == "B"].str.replace("B", "AB")
            labels[labels == "C"] = labels[labels == "C"].str.replace("C", "CDE")
            labels[labels == "D"] = labels[labels == "D"].str.replace("D", "CDE")
            labels[labels == "E1"] = labels[labels == "E1"].str.replace("E1", "CDE")
            labels[labels == "E2"] = labels[labels == "E2"].str.replace("E2", "CDE")
        elif transform == "AB_DE":
            mask_C = labels == "C"
            dataset = dataset.loc[~mask_C, :].reset_index(drop=True)
            labels = labels[~mask_C].reset_index(drop=True)
            labels[labels == "A"] = labels[labels == "A"].str.replace("A", "AB")
            labels[labels == "B"] = labels[labels == "B"].str.replace("B", "AB")
            labels[labels == "D"] = labels[labels == "D"].str.replace("D", "DE")
            labels[labels == "E1"] = labels[labels == "E1"].str.replace("E1", "DE")
            labels[labels == "E2"] = labels[labels == "E2"].str.replace("E2", "DE")
        else:
            raise ValueError("not a valid transformation")

    dataset = dataset[features]
    if visualize:
        print(labels.value_counts())
    return dataset, labels
```

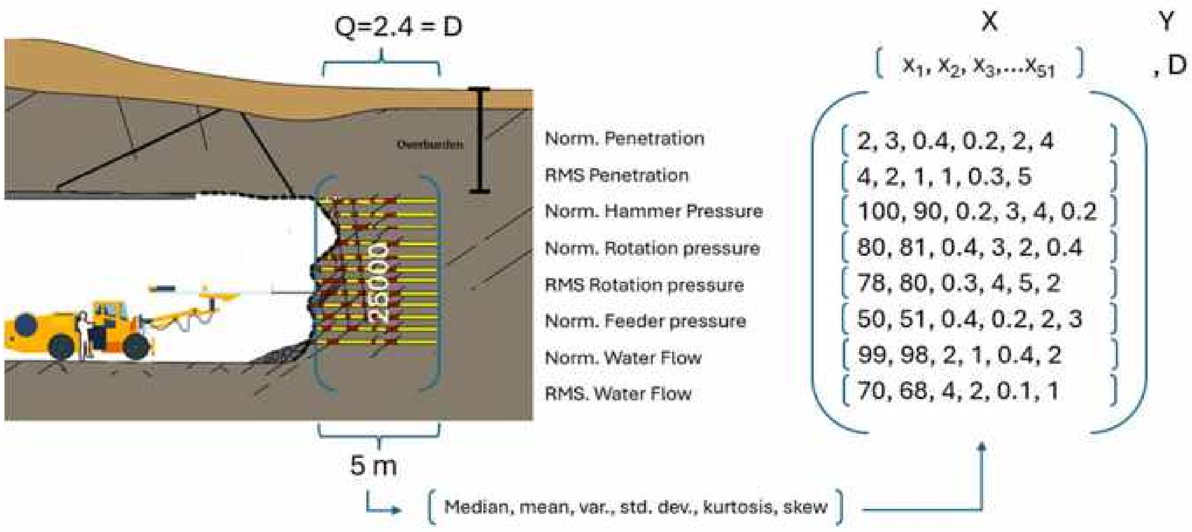
UiO : University of Oslo

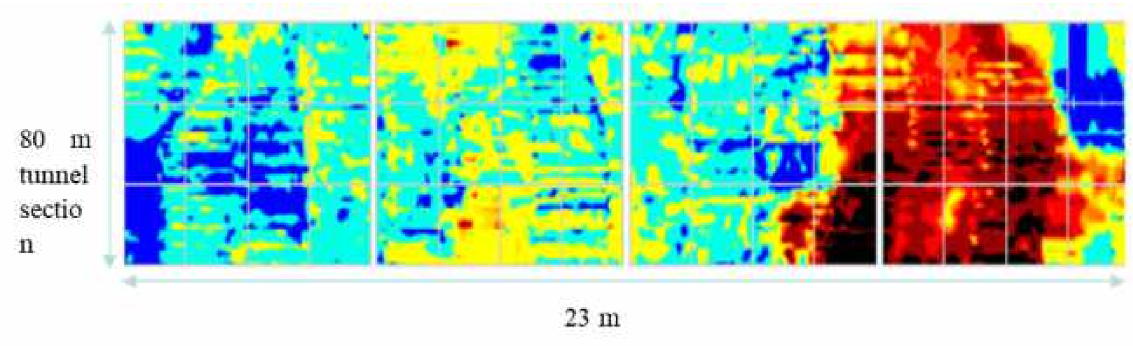
Dataset





Dataset

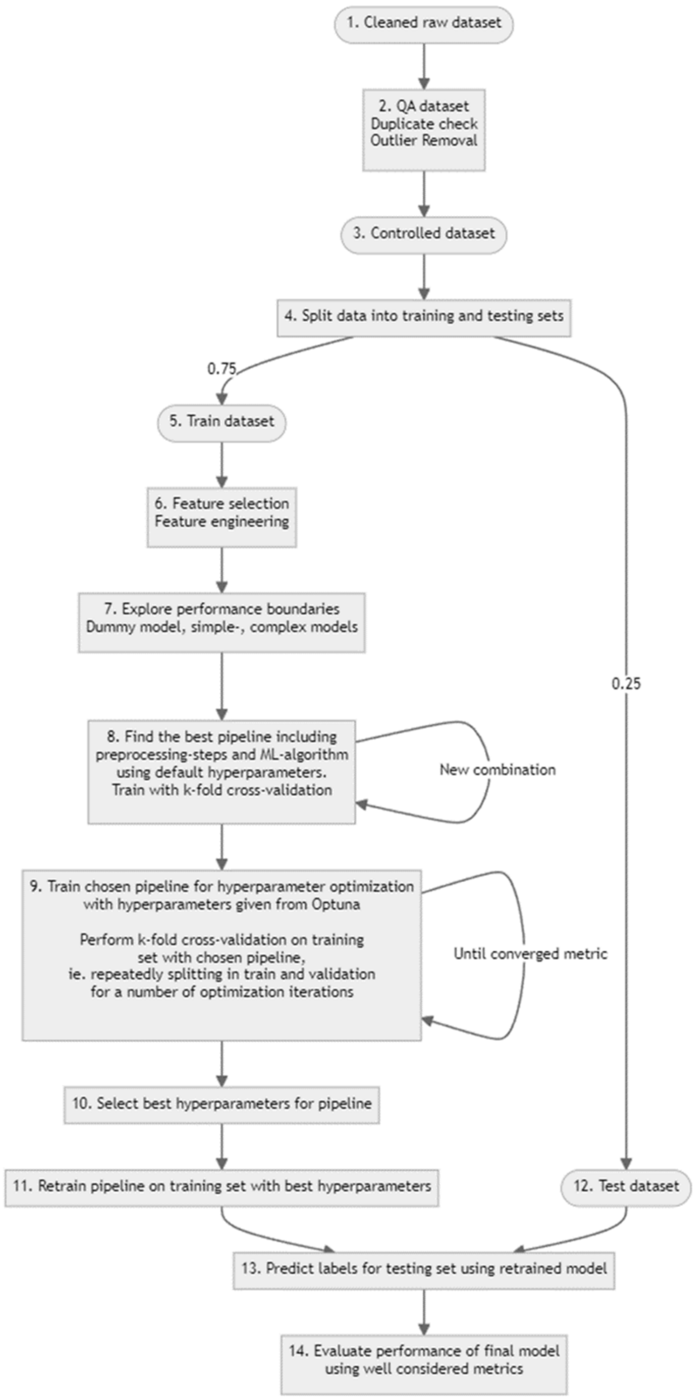

Q=2.4 = D
Overburden
25000
5 m
Norm. Penetration
RMS Penetration
Norm. Hammer Pressure
Norm. Rotation pressure
RMS Rotation pressure
Norm. Feeder pressure
Norm. Water Flow
RMS. Water Flow
X
Y
[x1, x2, x3,...x51] , D
[2, 3, 0.4, 0.2, 2, 4]
[4, 2, 1, 1, 0.3, 5]
[100, 90, 0.2, 3, 4, 0.2]
[80, 81, 0.4, 3, 2, 0.4]
[78, 80, 0.3, 4, 5, 2]
[50, 51, 0.4, 0.2, 2, 3]
[99, 98, 2, 1, 0.4, 2]
[70, 68, 4, 2, 0.1, 1]
[Median, mean, var., std. dev., kurtosis, skew]


80 m tunnel sectio n
23 m

UiO : University of Oslo

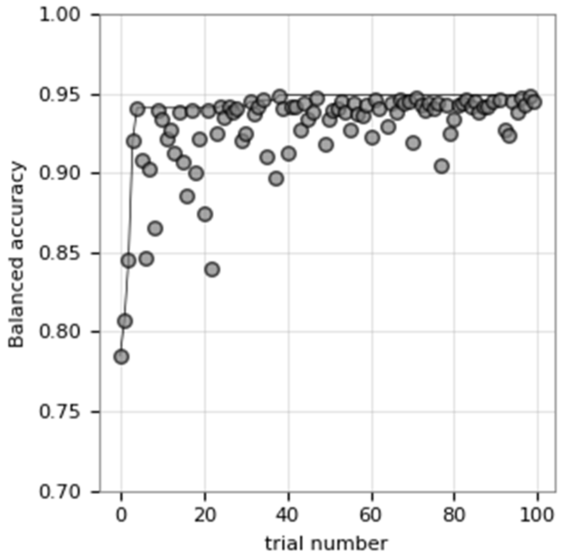
ML-based science

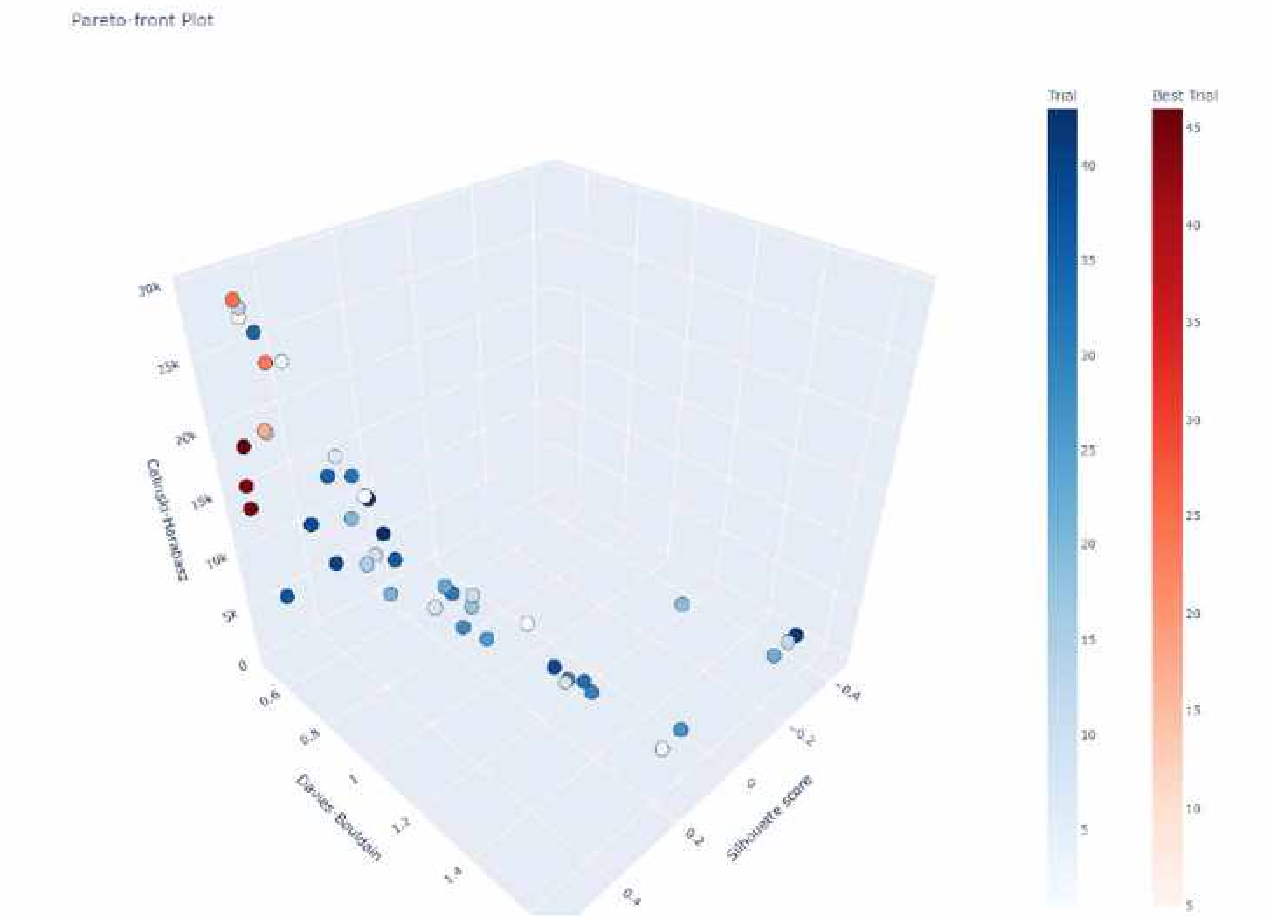
Objectives	Description of process
Code quality	This study aims to verify the hypothesis that rock types can be predicted using a trained machine-learning model applied to a labelled MWD dataset. The code serves as the detailed blueprint for this experiment; therefore, it must be understandable, clean, and well-structured. Code is read more often than it is written. We endeavoured to follow the main principles outlined by Wilson et al. [19], [20] and Martin [33]. We used meaningful variable names, modularised the code, used type annotations in Python to clarify the format of inputs and outputs, and provided documentation for each function. We used the industry standard auto-formatter, Black [34] to increase the code's readability and recognizability. We also set up test functions to detect errors, thus ensuring the quality of our experimentation and illustrating how a function operates.
Version controlling code and dataset	We organised a well-structured project and regularly committed the code using the version control system git to a private GitHub repository (accessible to reviewers), which will be made public upon the paper's acceptance. The dataset (model ready csv-files) was version-tracked using the Data Version Control (DVC) system [35] and quality-controlled while input-reading through Pandera [23] and shared on the scientific platform Zenodo [36].
Controlling programming environment	We leveraged Poetry [37], an environment and package handling system, to manage dependencies. Poetry automatically generates a lock file describing all packages and their corresponding versions. The Python version used in this project was specified in a *.python-version* file and managed using the Pyenv tool [38], simplifying the process of downloading and switching between Python versions. Notable differences exist between Python versions, such as Python 3.9 (introduction of match-case statements), Python 3.10 (advanced type annotations), and Python 3.11 (performance enhancements), underscoring the importance of this process.
Configuration of parameters	Configuration values were controlled and type-parsed with Pydantic (and enhanced with tab completion) [35]. Experiment results were thoroughly organised and saved for all experiments using the Hydra [39] and MLflow [40] systems. Hydra helps avoid the pitfall of embedding "magic numbers" within the code and enables swift experimentation prototyping from the terminal.
Experiment tracking	MLflow gives you an overview and a log of all experiments with results in a clean web-based view. Each model step (finding pipeline, hyperparameter optimisation, evaluation, final training) was grouped in experiments in Mlflow for easy comparison of experiment performance.
Orchestrating experiments	In our research, we integrated Hydra with GNU Make [41], a widely used automation tool, to execute scientific experiments efficiently. Hydra manages and organises diverse experiment configurations, enabling flexible and scalable setups. GNU Make, encapsulated in a Makefile, orchestrates these experiments, ensuring reproducibility and efficiency. This methodology not only streamlines the experiment process but also facilitates ease of replication for other researchers, embodying the principles of open and reproducible science.
Control randomness	Seed values were established to control the randomness in data splitting and algorithms. Unless clearly stated, seeding has been used in all experiments to be able to compare results. However, seeding was regularly turned off to explore the spread of results.
Control operating	Employing all the processes mentioned above enables the reproducibility of our research results in nearly all instances, provided the same dataset is used. However, an exception could be made when the hardware controlling software, such as CPU and GPU drivers,



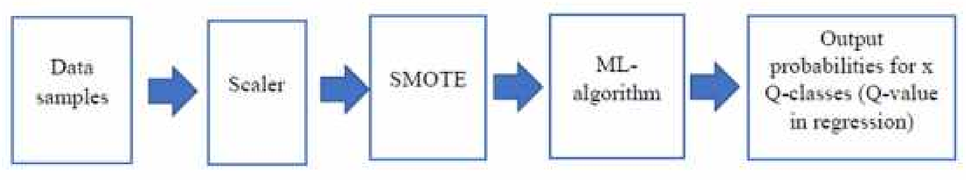
The principles and pipeline in the experimentation process for machine learning are presented in an online public presentation: https://prezi.com/view/chJ8Djt4GKjeAdFiGvAl/

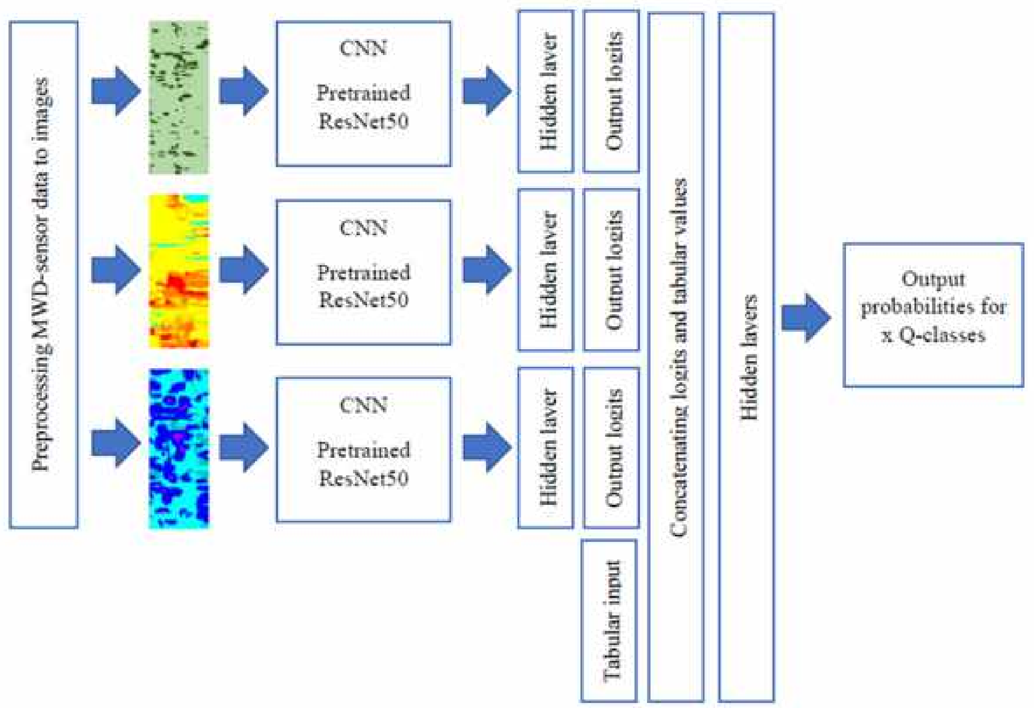
The power of smart optimisation



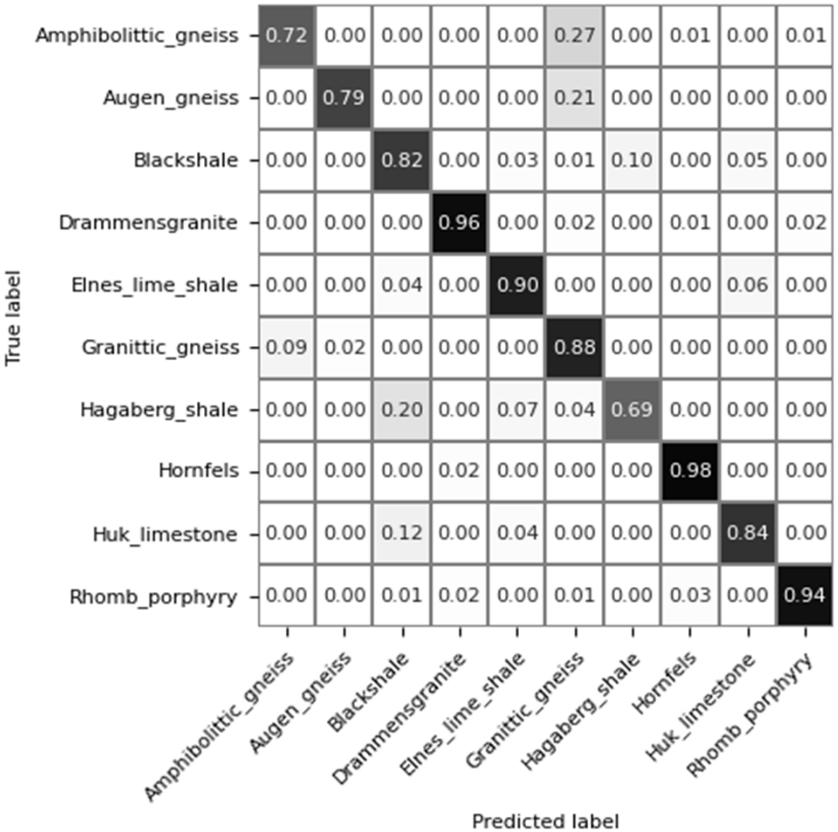


Architecture – supervised prediction





Supervised prediction models



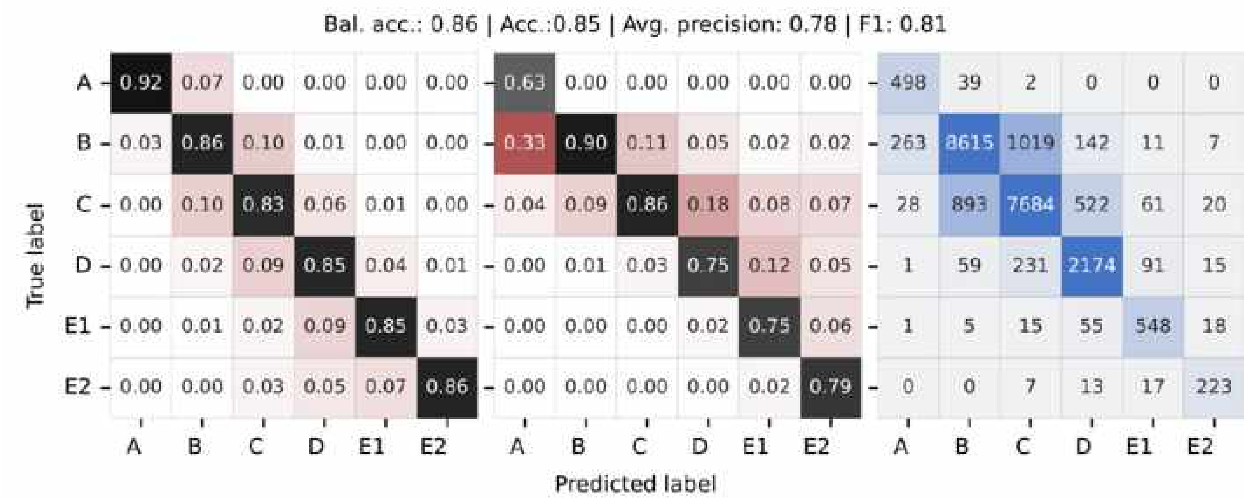
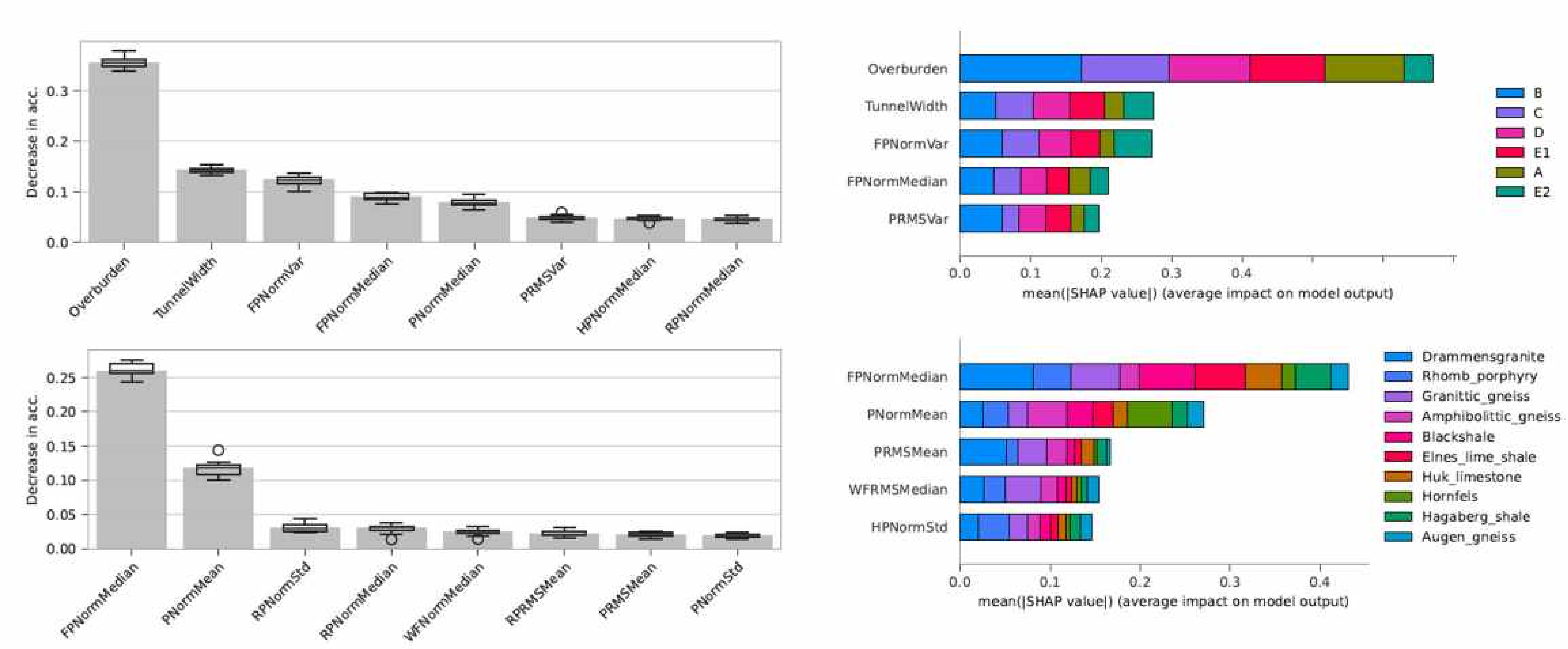


Figure 7. Confusion matrix for a Voting Classifier optimised for recall and trained with 5-fold cross-validation.

Explainability



Explainability

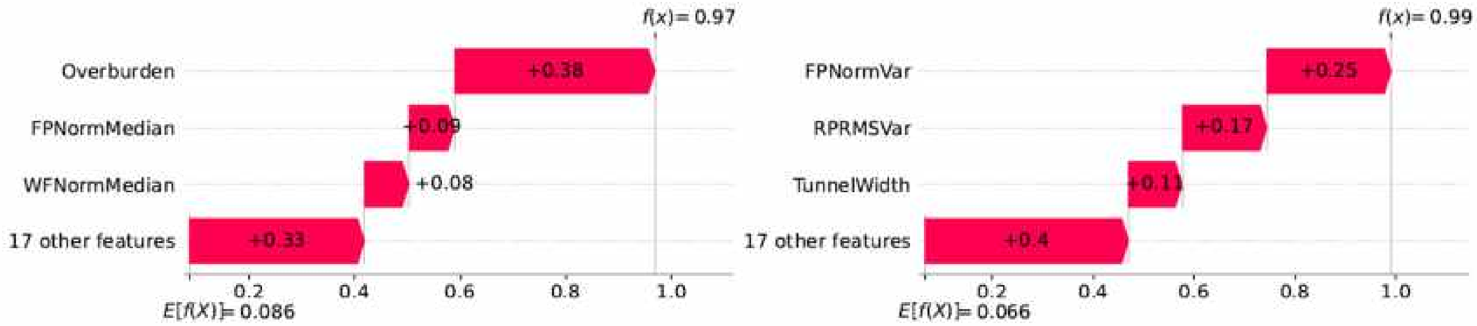


Fig. 3. Waterfall plots of Shapley values for the three most important features in predicting a sample of **a** Q-class A, and **b** Q-class E2.

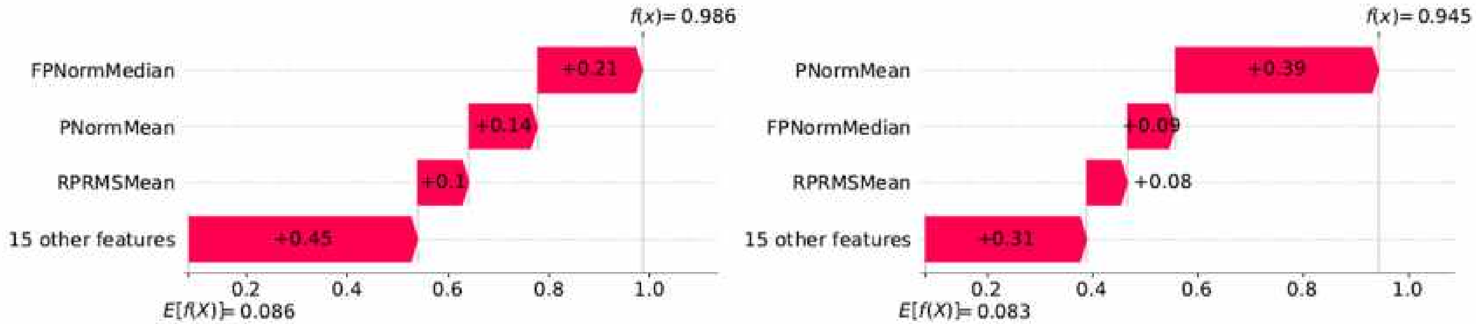


Fig. 4. Waterfall plots of Shapley values for the three most important features in predicting a sample of **a** rock type Blackshale, and **b** rock type Hornfels.

Explainability

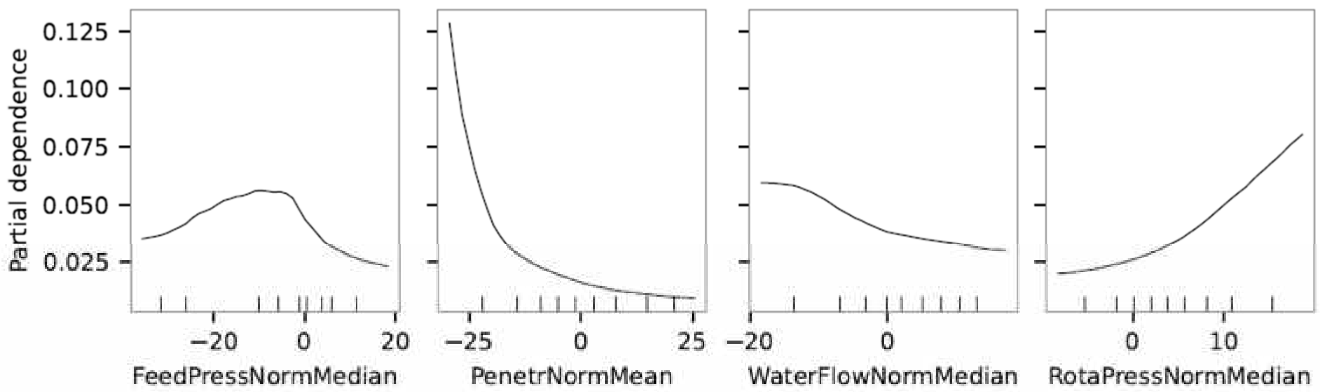


Fig. 7. Partial dependency plots for four significant features identifying rocktype, examplified for Hornfels.

Clustering

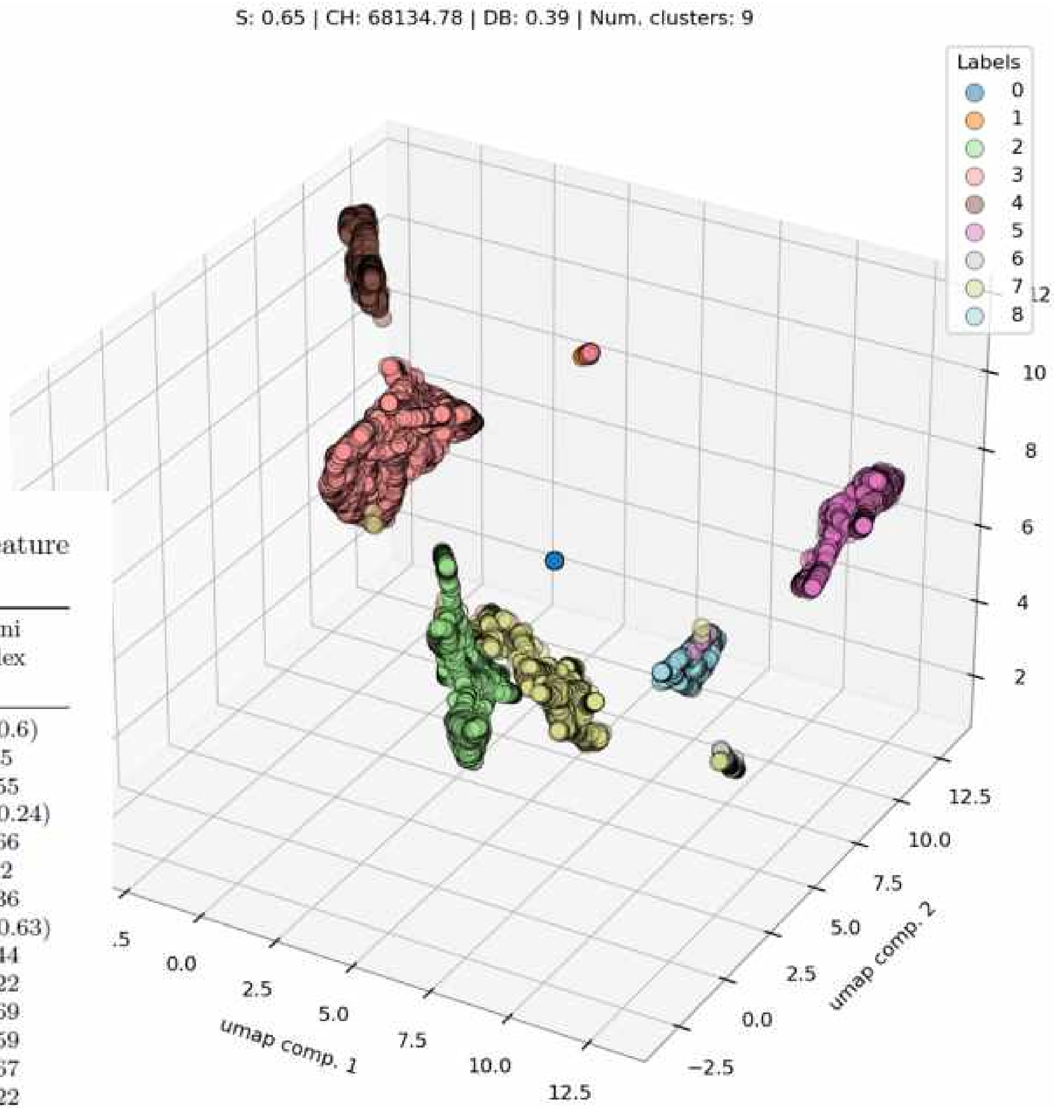


Table 2: Summary of clustering results for four different feature sets, grouped by feature sets. Scores for default algorithm parameters in parenthesis

Id	Feature set	Num. features	Dim. red. alg.	Cluster alg.	Num. clusters	Num. dim. red. comp.	Num. not clustered samples	Gini index
0	all	50	umap	hdbscan	9(956)	12(2)	0(6140)	0.5(0.6)
1	all	50	umap	hdbscan	9	15	23	0.5
2	mwd	48	umap	aggl. clust.	6	7	0	0.55
3	mwd	48	umap	aggl. clust.	7(6)	6(2)	0(0)	0.57(0.24)
4	mwd	48	umap	hdbscan	5	12	23	0.66
5	mwd	48	umap	kmeans	7	10	0	0.2
6	mwd	48	umap	kmeans	3	4	0	0.36
7	mwd	48	umap	hdbscan	11(838)	3(2)	55(6053)	0.62(0.63)
8	mwd	48	umap	hdbscan	13	12	1195	0.44
9	mwd	48	pca	kmeans	10	2	0	0.22
10	mwd	48	umap	hdbscan	6	2	22	0.69
11	mwd	48	pca	hdbscan	3	5	1842	0.59
12	mwd	48	None	hdbscan	3	—	1	0.67
13	mwd_rock	30	pca	kmeans	6	2	0	0.22
14	mwd_rock	30	umap	hdbscan	11	4	175	0.68
15	mwd_rock	30	umap	hdbscan	9	15	1014	0.64
16	mwd_median	8	umap	hdbscan	6	11	23	0.74

RL – the full tunnel cycle

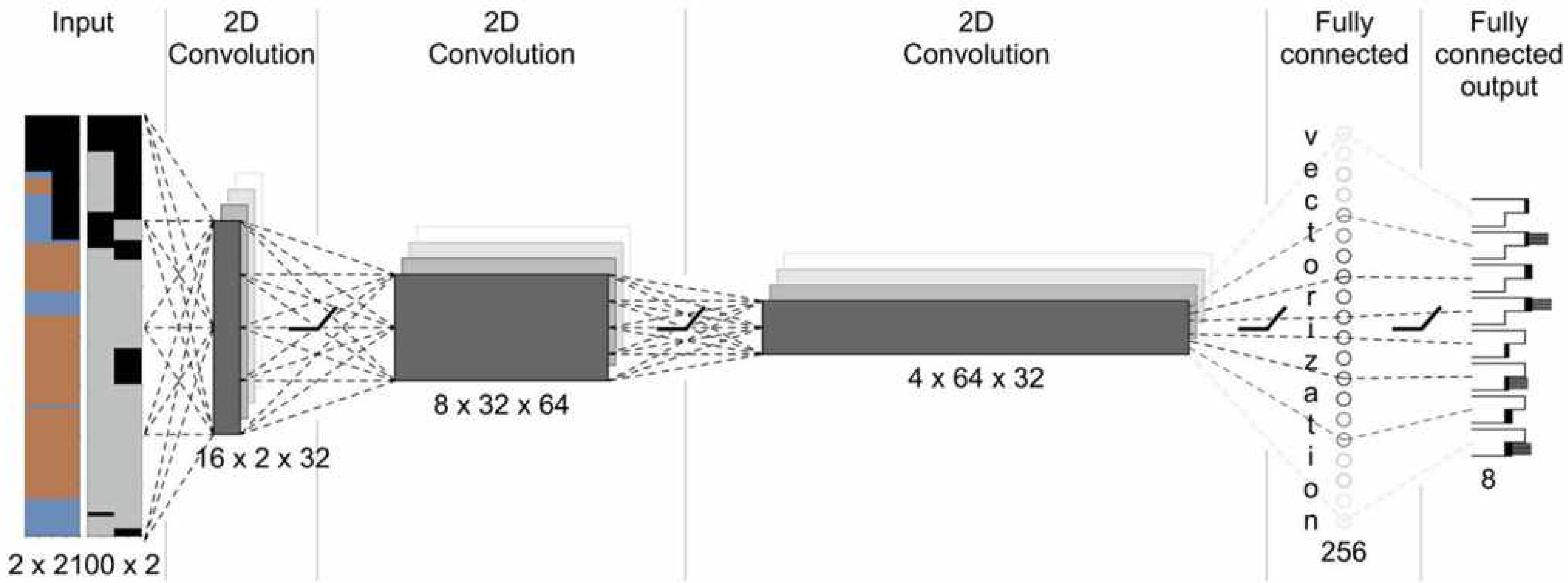


Fig. 5. Schematic representation of the DQN agent's ANN architecture. Note the visualization of rockmass-matrix and the support-matrix to the left. The numbers below each layer are the respective shape of the layer's weights. Dashed connection lines between layers are only for illustrational purposes. Symbols at the output layer represent the eight possible actions (ordered as in Table 3) that are chosen via Q-values by the agent.

RL – the full tunnel cycle

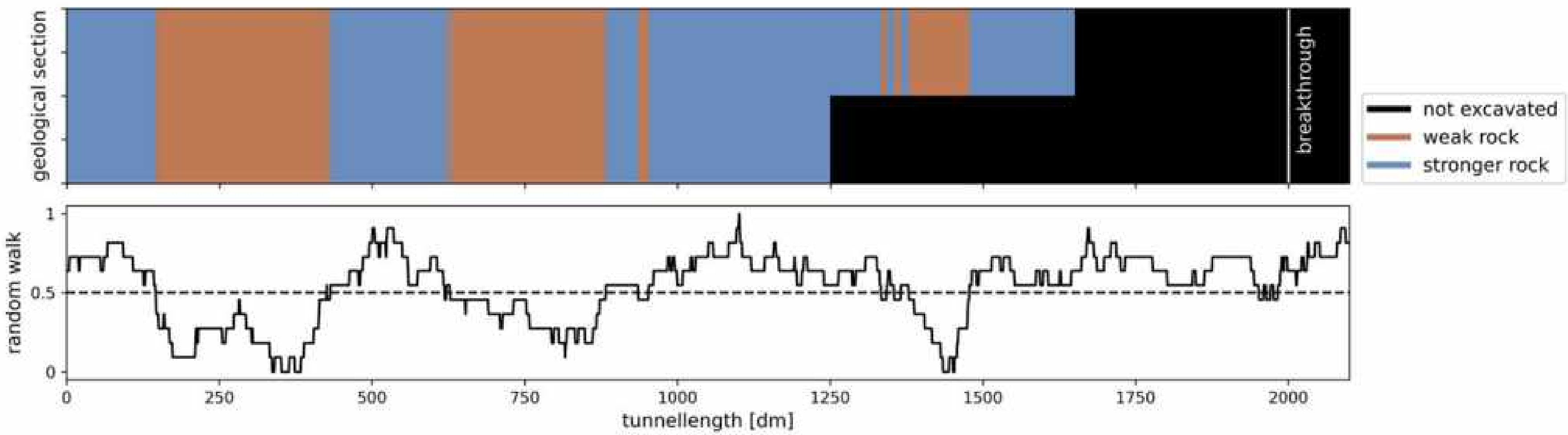
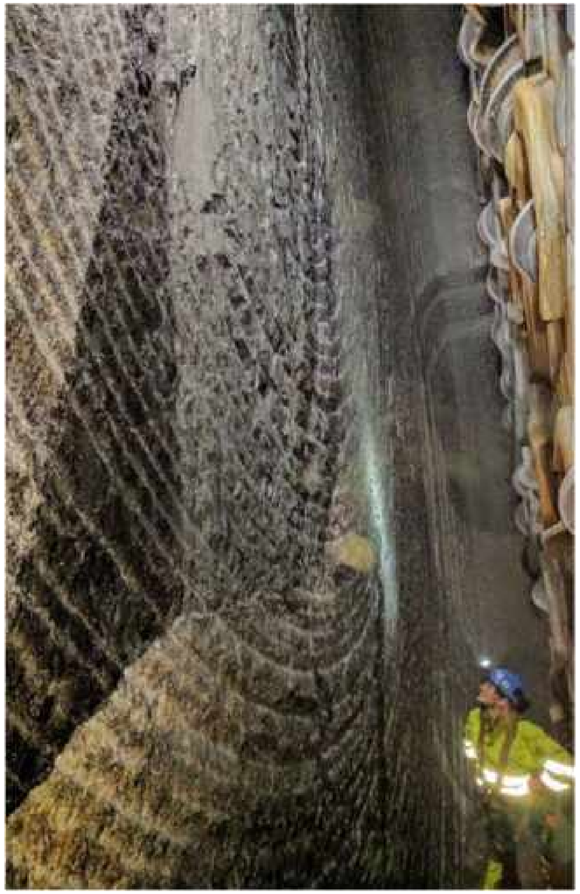


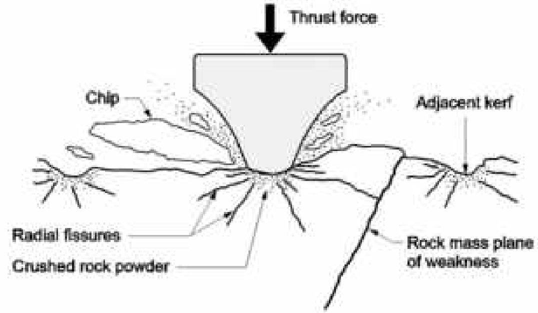
Fig. 4. Top row: an exemplary unique geological section, where brown indicates weak (gt1) and blue stronger rock (gt2). The positions of the top heading and bench are at 165.0 m and 125.0 m respectively. Bottom row: the random walk that is used to generate the geological section. Values above 0.5 are converted to gt2 and below to gt1. Note that the x-axis is the tunnel length in decimeters which corresponds to the number of datapoints of the random walk.

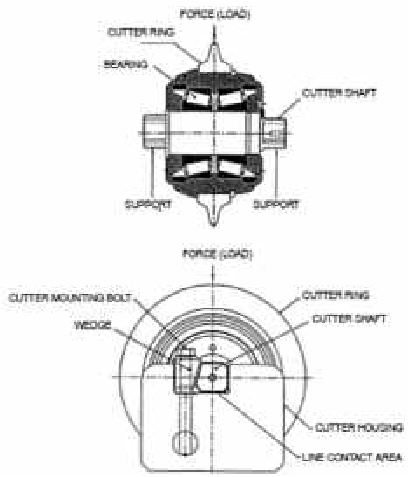
RL – predictive maintenance

Wearing down the cutter disks

Today’s maintenance decisions: human subjectiveness



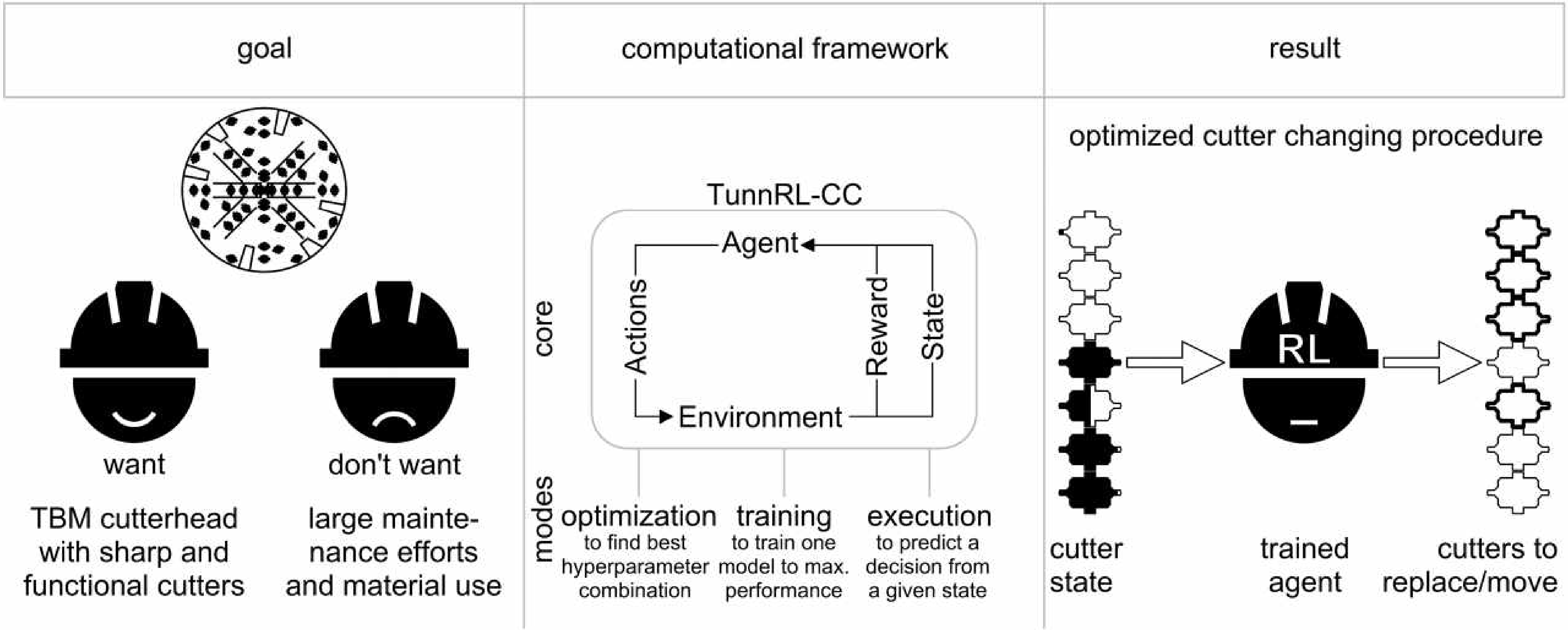




Images from Bane NOR and NFF

UiO University of Oslo NGI DIGITAL

TunnRL – CC – An automated decision system

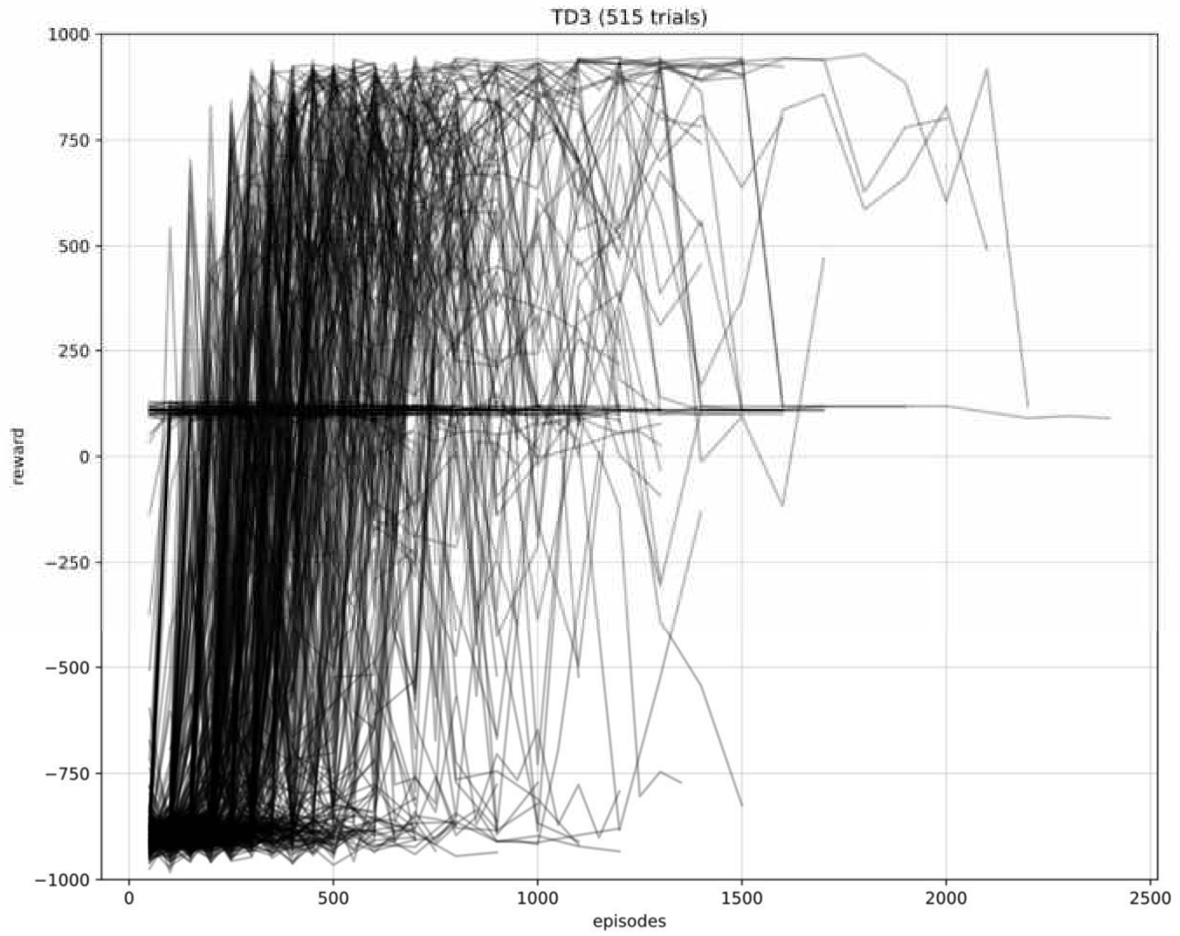


Agent performance

Algorithm	Maximum Reward	Episode num max reward	Number of trials for the algorithm	Avg. replaced cutters	Avg. moved cutters	Avg. broken cutters
TD3 (off)	945	1264	515	0.028	1.66	1.348
DDPG (off)	879	400	393	0.002	2.13	1.346
A2C (on)	650	3448	324	29.8	11.2	0
PPO (on)	637	5296	293	35.04	1.23	0.775
SAC (on)	205	468	41	0.862	34.3	0.14

TD3 – Twin Delayed DDPG

(DDPG – Deep deterministic policy gradient)



UiO : University of Oslo

Excavating 540 m tunnel (1 stroke = 1.8 m)