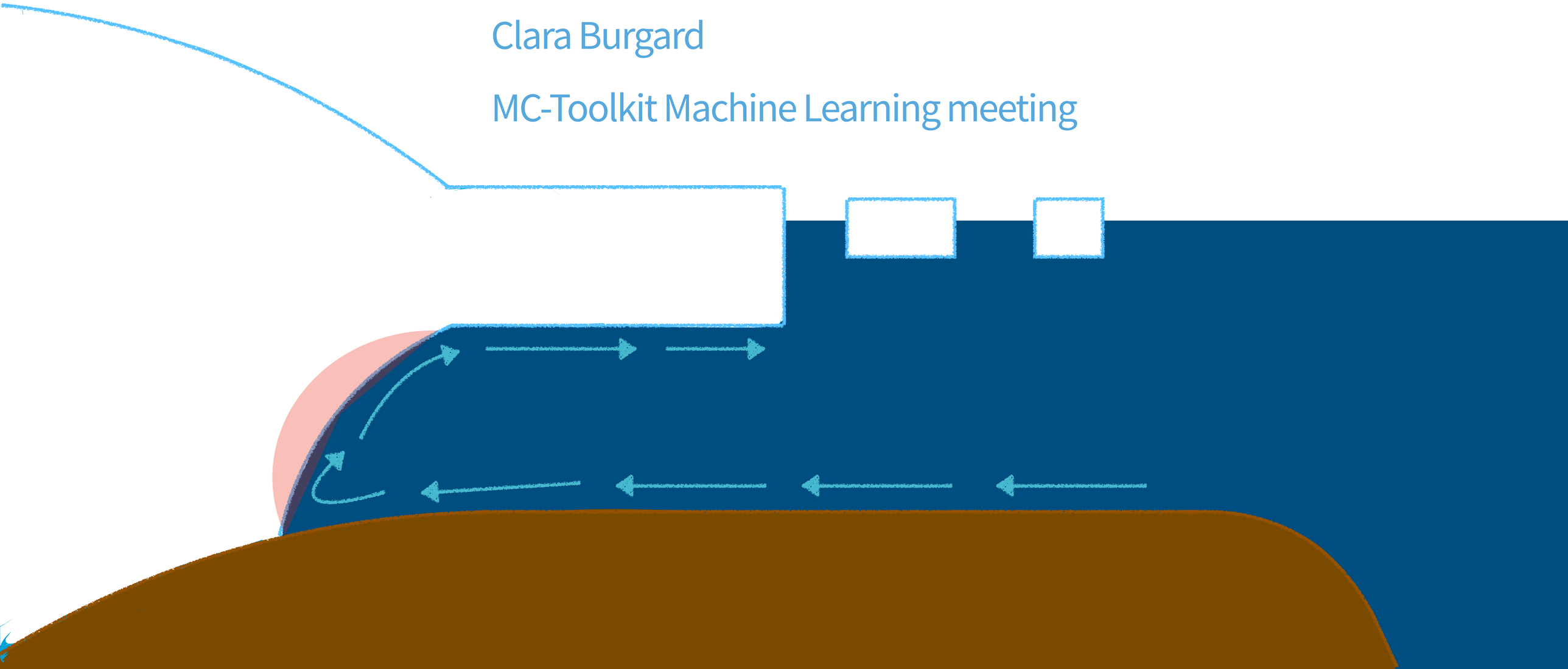
Representing the ocean-induced sub-shelf melt in Antarctica with a neural network?

Clara Burgard

MC-Toolkit Machine Learning meeting

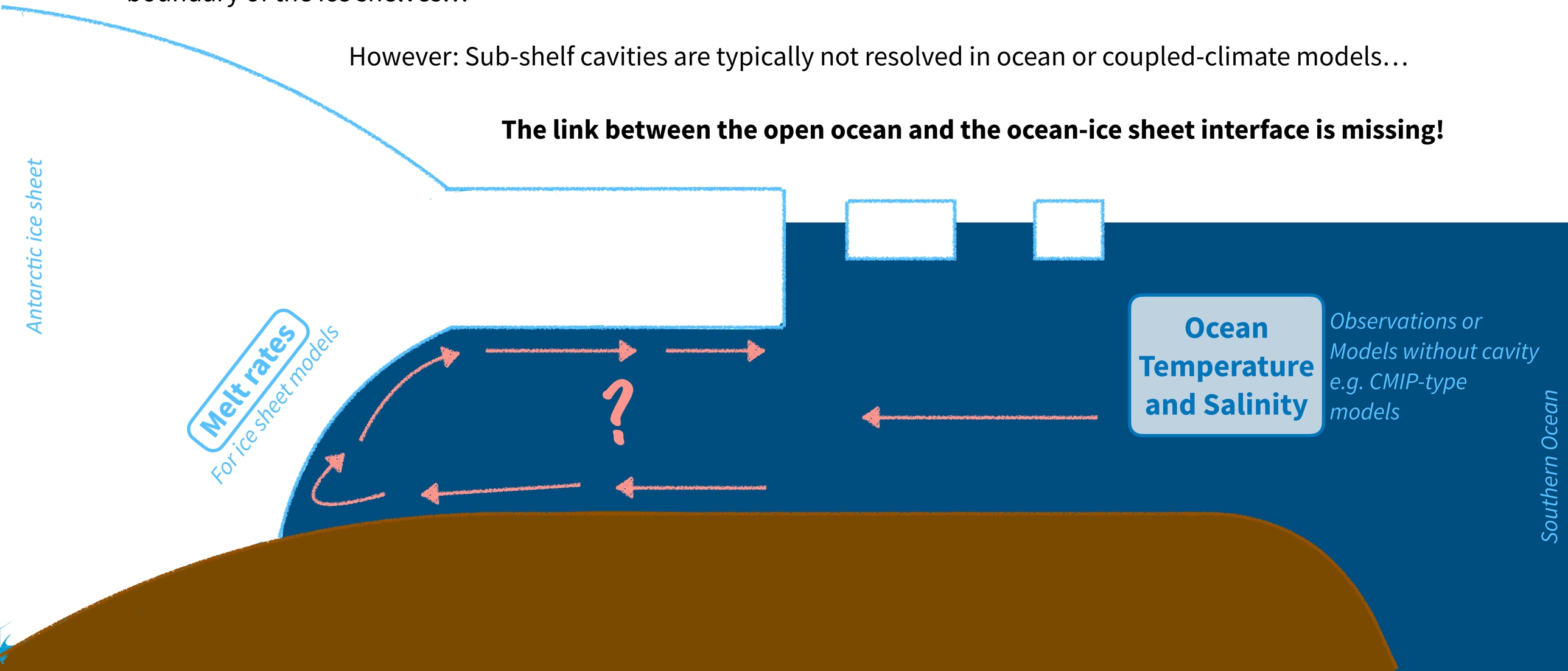


The problem: Representing ocean basal melting in (uncoupled) ice sheet models

Ice sheet models need information about the ocean water interacting with the ice at the lower boundary of the ice shelves…

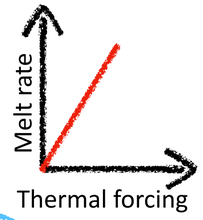
However: Sub-shelf cavities are typically not resolved in ocean or coupled-climate models…

The link between the open ocean and the ocean-ice sheet interface is missing!



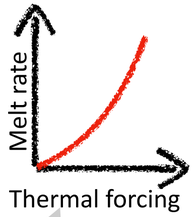
Until now: Assess and improve existing ocean basal melting parameterisations for ice sheet models

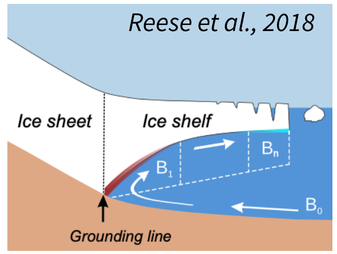
We assess and tune existing sub-shelf melt parameterisations in a circum-Antarctic approach.



Linear function of thermal forcing
[Beckmann & Goosse (2002)]

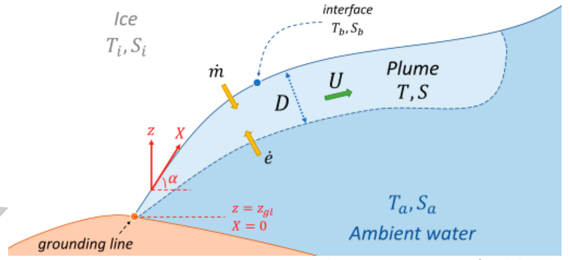
Quadratic function of thermal forcing
[De Conto & Pollard (2016), Favier et al. (2019)]





Box model
[Olbers & Hellmer (2010), Reese et al. (2018)]

Plume parameterisation
[Jenkins (1991), Lazeroms et al. (2018 & 2019)]



Lazeroms et al., 2019

Antarctic ice sheet

Melt rates
For ice sheet models

Melt parameterisation

Ocean Temperature and Salinity

Observations or Models without cavity e.g. CMIP-type models

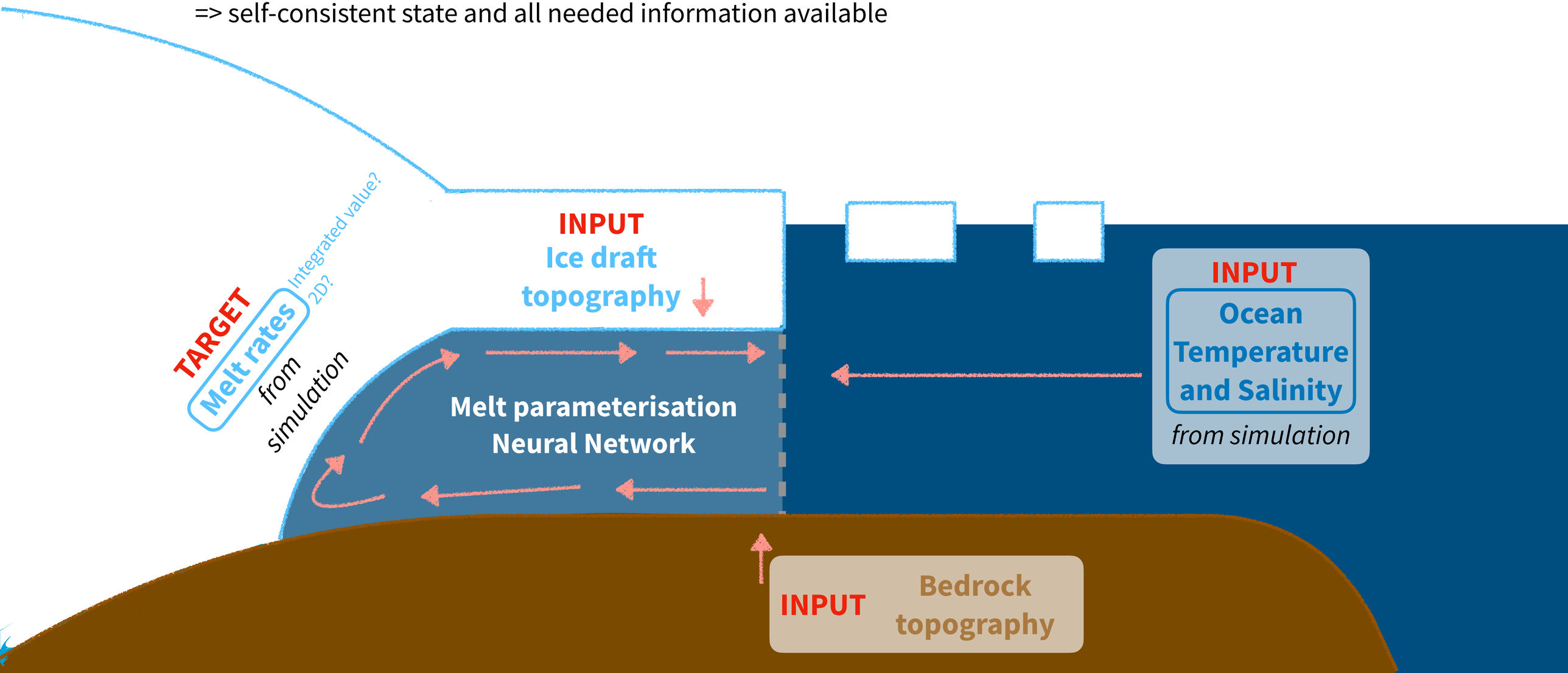
Southern Ocean

Idea: What I wanted to try with neural networks

“Perfect model” approach

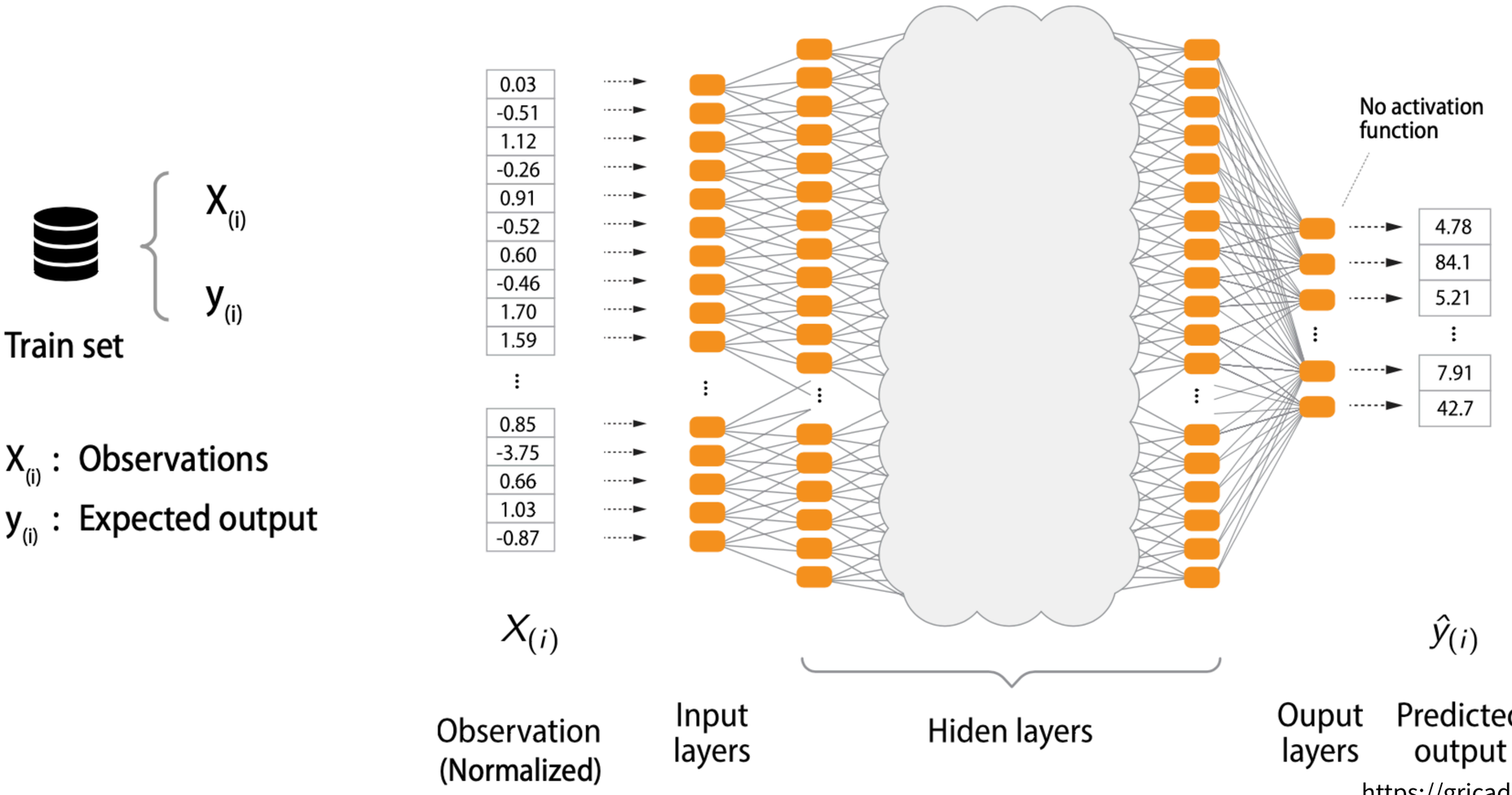
We could use circum-Antarctic ocean simulation (resolving cavities) or ideal experiments (MISOMIP) as a virtual reality to train a neural network

=> self-consistent state and all needed information available



How to approach the problem?

Regression with a DNN

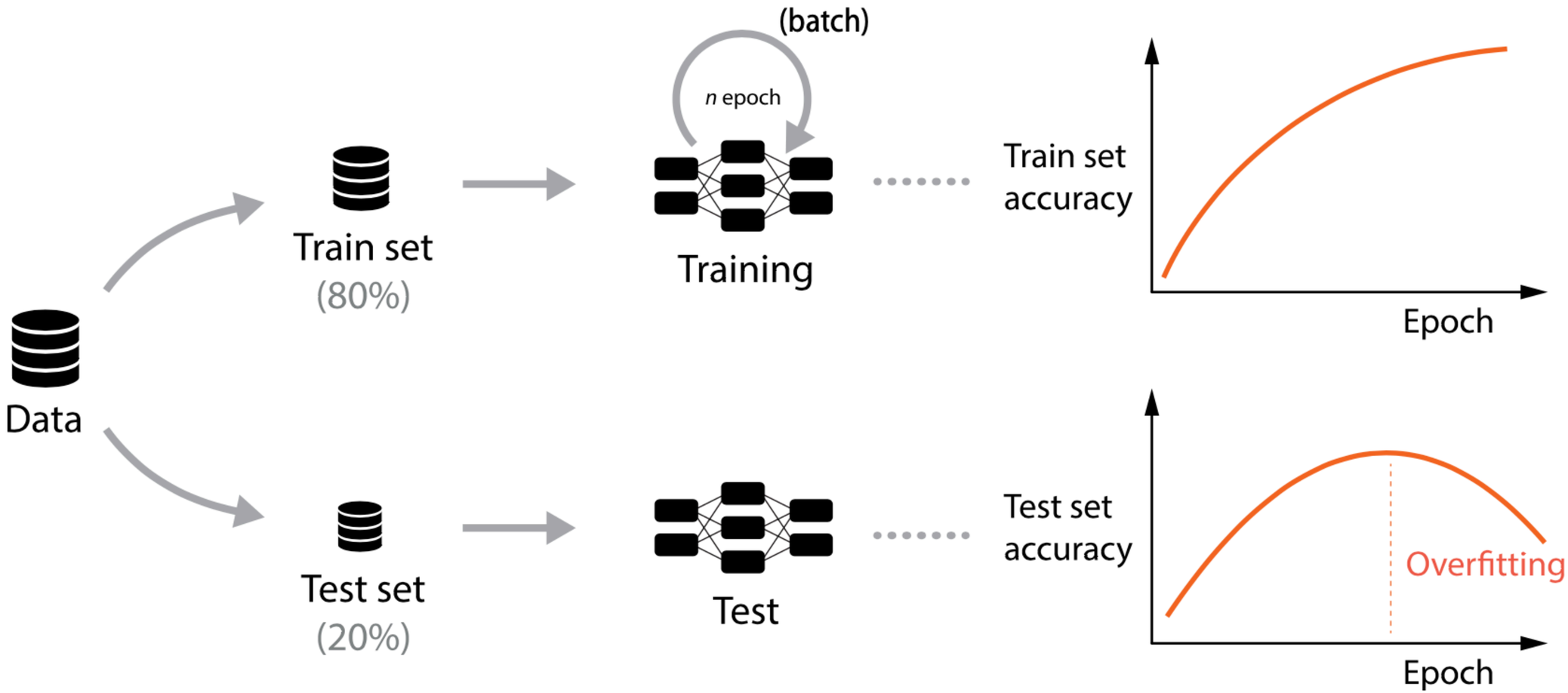


https://gricad-gitlab.univ-grenoble-alpes.fr/talks/fidle

Source: FIDLE course, CNRS-SARI/Resinfo/DEVLOG

How to train the model?

Training process - general



https://gricad-gitlab.univ-grenoble-alpes.fr/talks/fidle

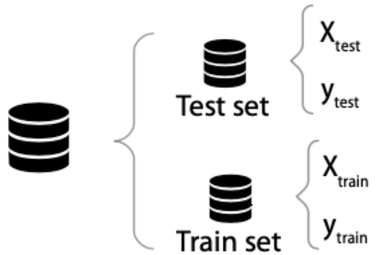
Source: FIDLE course, CNRS-SARI/Resinfo/DEVLOG

Implementation steps

Step 1 - Import and init

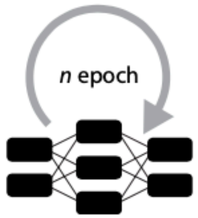
Step 2 - Retrieve data

Step 3 - Preparing the data

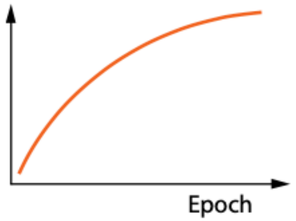


Step 4 - Build a model

Step 5 - Train the model



Step 6 - Evaluate



https://gricad-gitlab.univ-grenoble-alpes.fr/talks/fidle

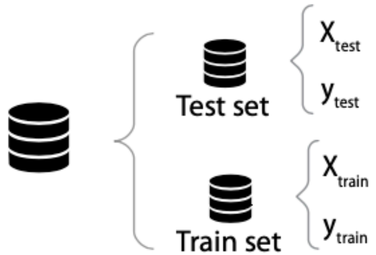
Source: FIDLE course, CNRS-SARI/Resinfo/DEVLOG

Implementation steps

Step 1 - Import and init

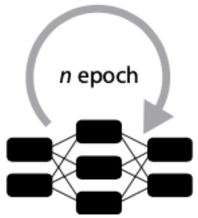
Step 2 - Retrieve data

Step 3 - Preparing the data

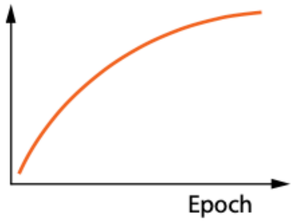


Step 4 - Build a model

Step 5 - Train the model



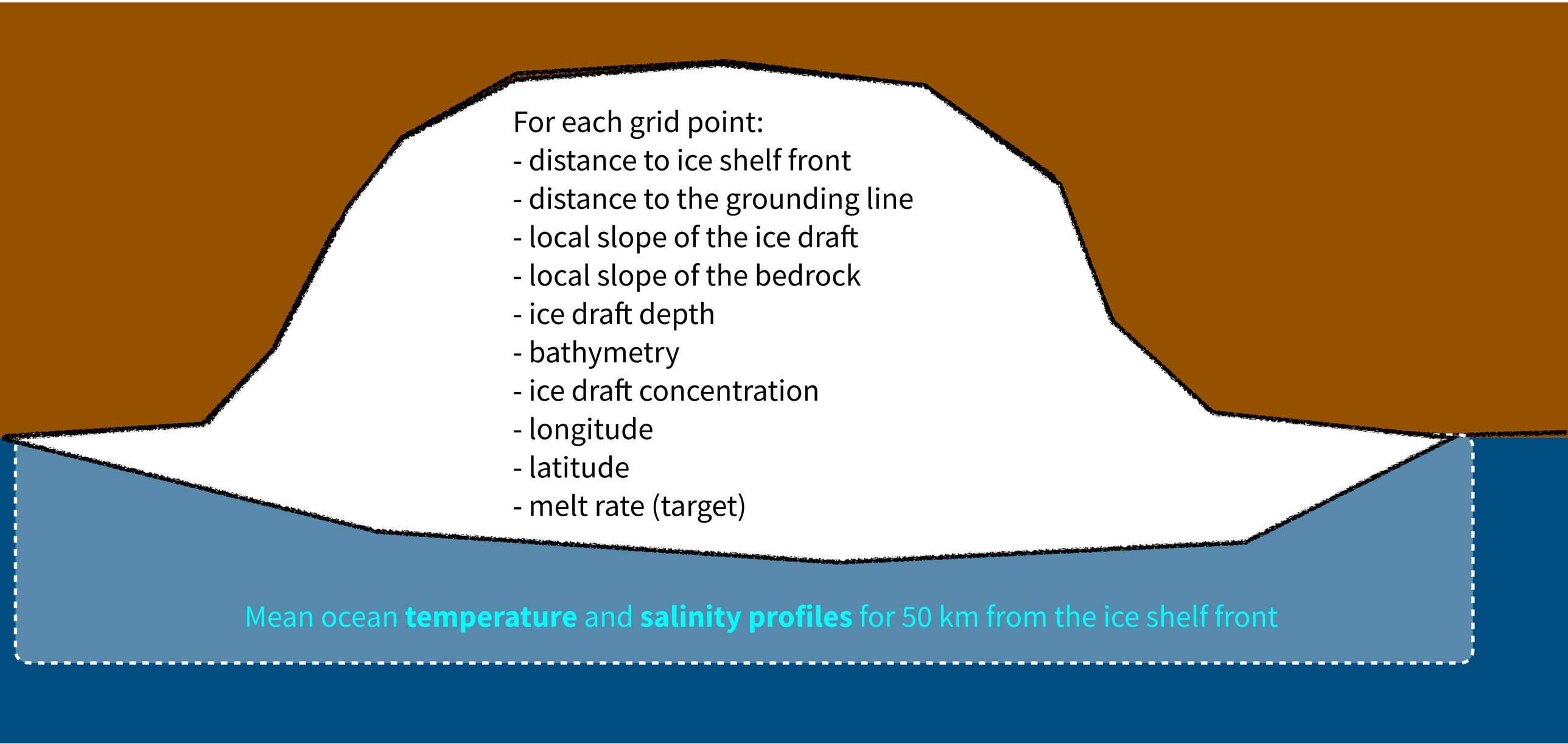
Step 6 - Evaluate



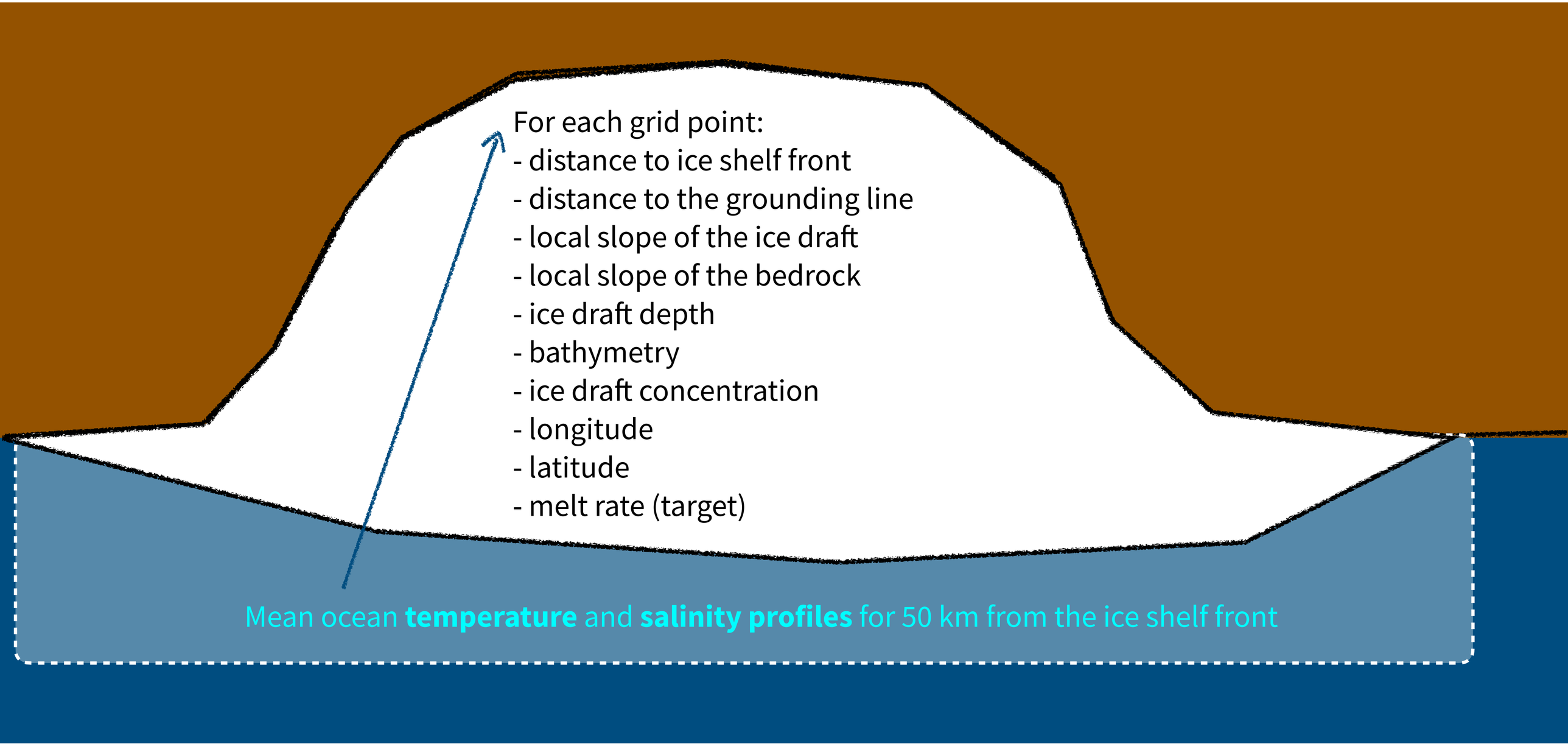
https://gricad-gitlab.univ-grenoble-alpes.fr/talks/fidle

Source: FIDLE course, CNRS-SARI/Resinfo/DEVLOG

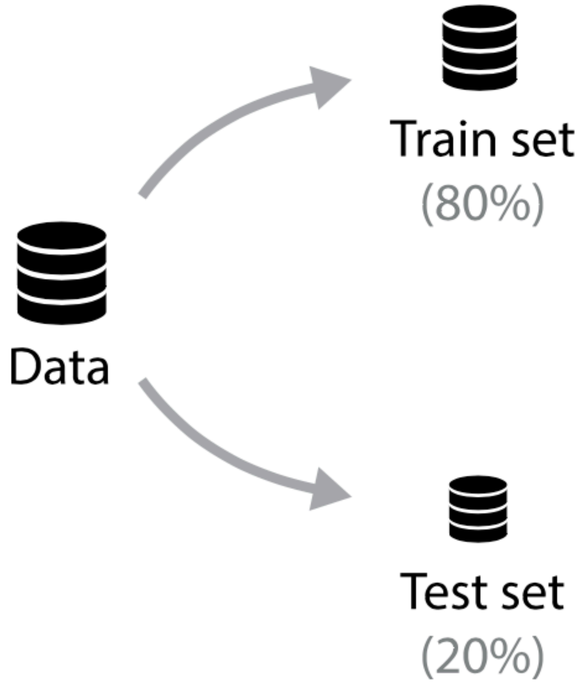
Preparing the data



Preparing the data



Preparing the data



https://gricad-gitlab.univ-grenoble-alpes.fr/talks/fidle

Source: FIDLE course, CNRS-SARI/Resinfo/DEVLOG

Implementation steps

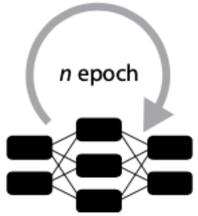
Step 1 - Import and init

Step 2 - Retrieve data

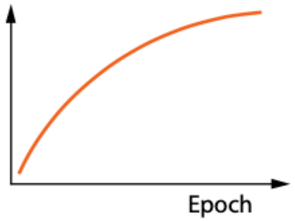
Step 3 - Preparing the data

Step 4 - Build a model

Step 5 - Train the model



Step 6 - Evaluate



https://gricad-gitlab.univ-grenoble-alpes.fr/talks/fidle

Source: FIDLE course, CNRS-SARI/Resinfo/DEVLOG

Build the model

```python
def get_model_v1(shape):

    model = keras.models.Sequential()
    model.add(keras.layers.Input(shape, name="InputLayer"))
    model.add(keras.layers.Dense(32, activation='relu', name='Dense_n1'))
    model.add(keras.layers.Dense(64, activation='relu', name='Dense_n2'))
    model.add(keras.layers.Dense(32, activation='relu', name='Dense_n3'))
    model.add(keras.layers.Dense(1, name='Output'))

    model.compile(optimizer = 'adam',
                  loss      = 'mse',
                  metrics   = ['mae', 'mse'] )
    return model
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
Dense_n1 (Dense)             (None, 32)                5216
_________________________________________________________________
Dense_n2 (Dense)             (None, 64)                2112
_________________________________________________________________
Dense_n3 (Dense)             (None, 32)                2080
_________________________________________________________________
Output (Dense)               (None, 1)                 33
=================================================================
Total params: 9,441
Trainable params: 9,441
Non-trainable params: 0
_________________________________________________________________
```

Implementation steps

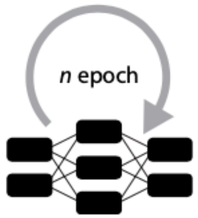
Step 1 - Import and init

Step 2 - Retrieve data

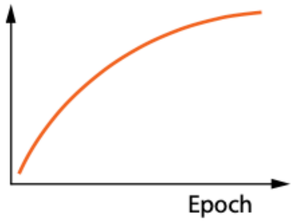
Step 3 - Preparing the data

Step 4 - Build a model

Step 5 - Train the model



Step 6 - Evaluate



https://gricad-gitlab.univ-grenoble-alpes.fr/talks/fidle

Source: FIDLE course, CNRS-SARI/Resinfo/DEVLOG

Train the model

```
history = model.fit(x_train_arr,
                    y_train_arr,
                    epochs          = 60,
                    batch_size      = 10,
                    verbose         = 1,
                    validation_data = (x_test_arr, y_test_arr))
```

Implementation steps

Step 1 - Import and init

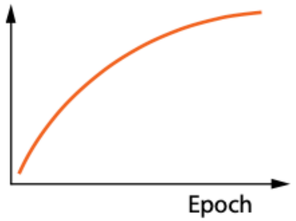
Step 2 - Retrieve data

Step 3 - Preparing the data

Step 4 - Build a model

Step 5 - Train the model

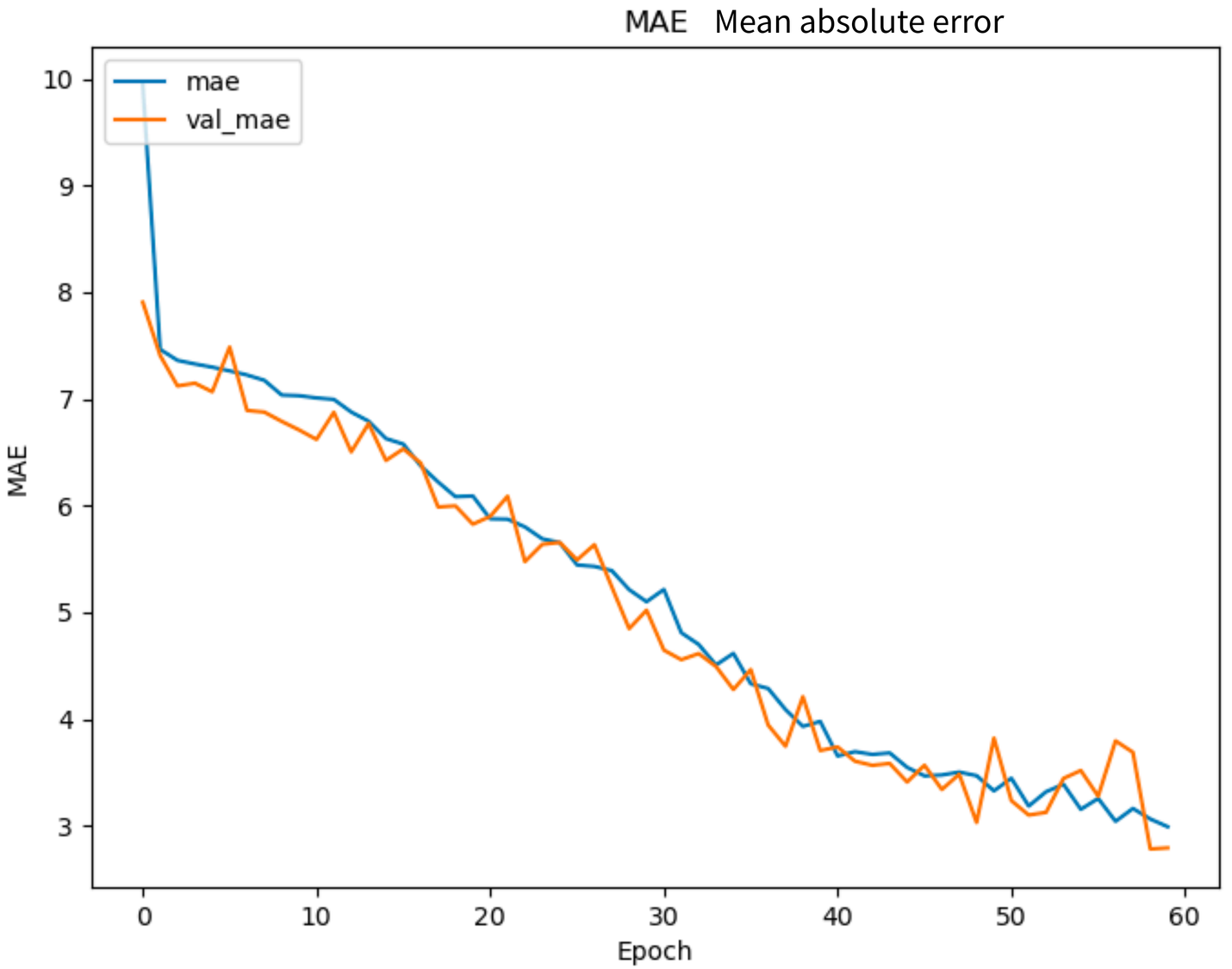
Step 6 - Evaluate



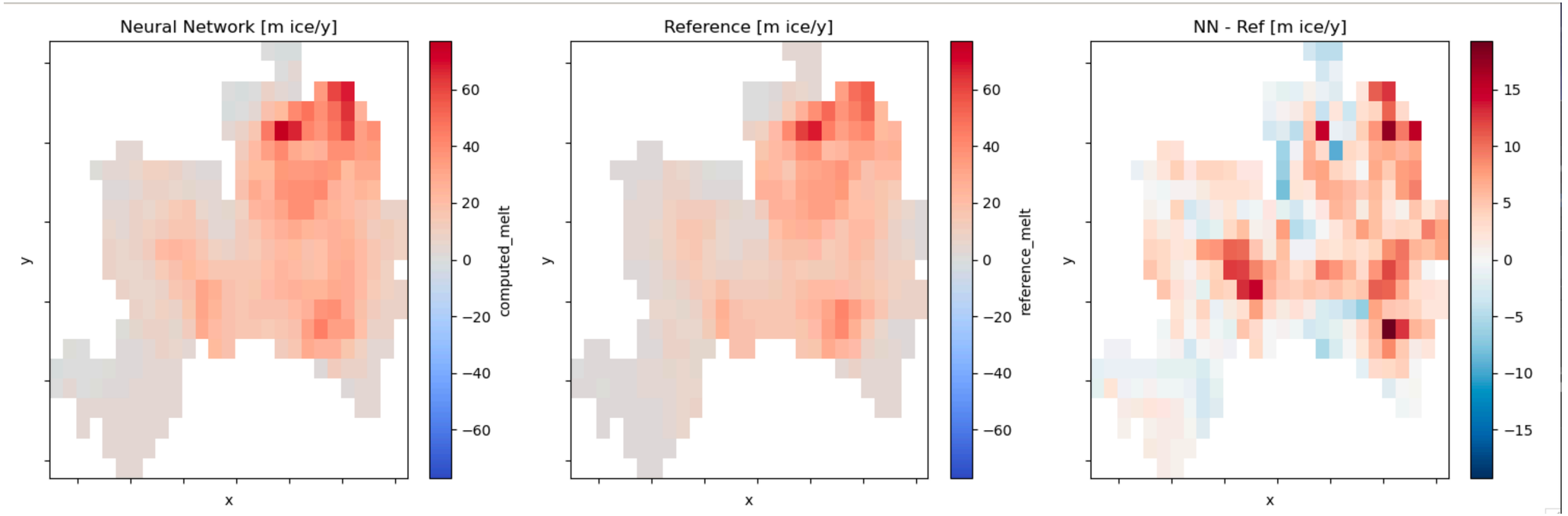
https://gricad-gitlab.univ-grenoble-alpes.fr/talks/fidle

Source: FIDLE course, CNRS-SARI/Resinfo/DEVLOG

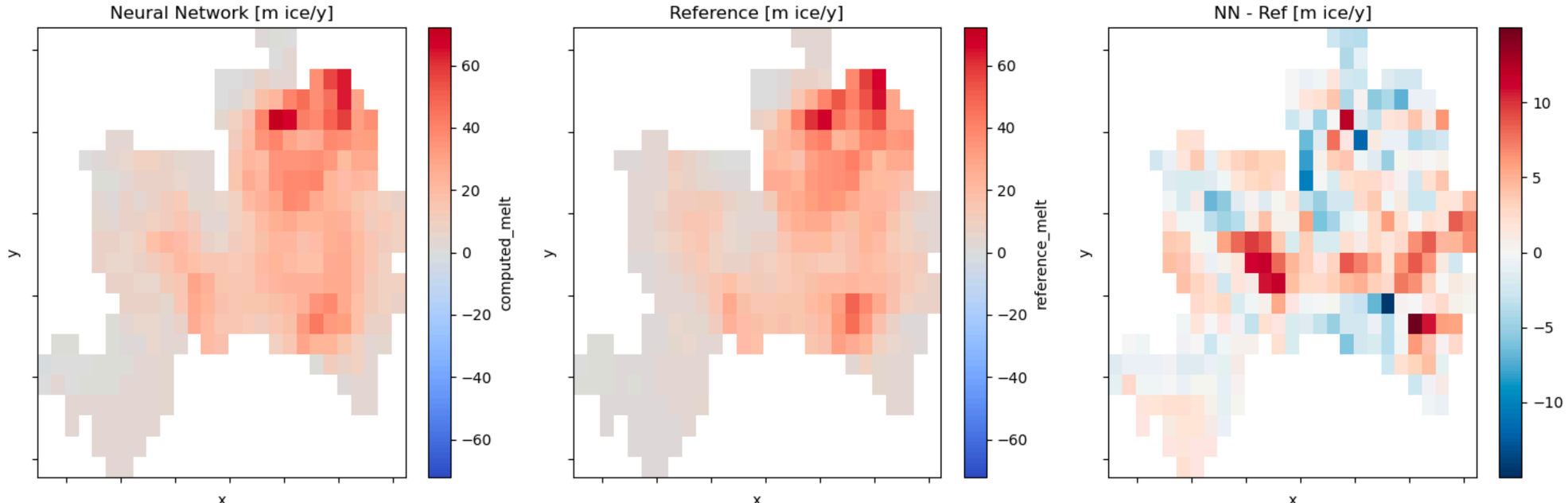
Evaluate the model



Make a prediction



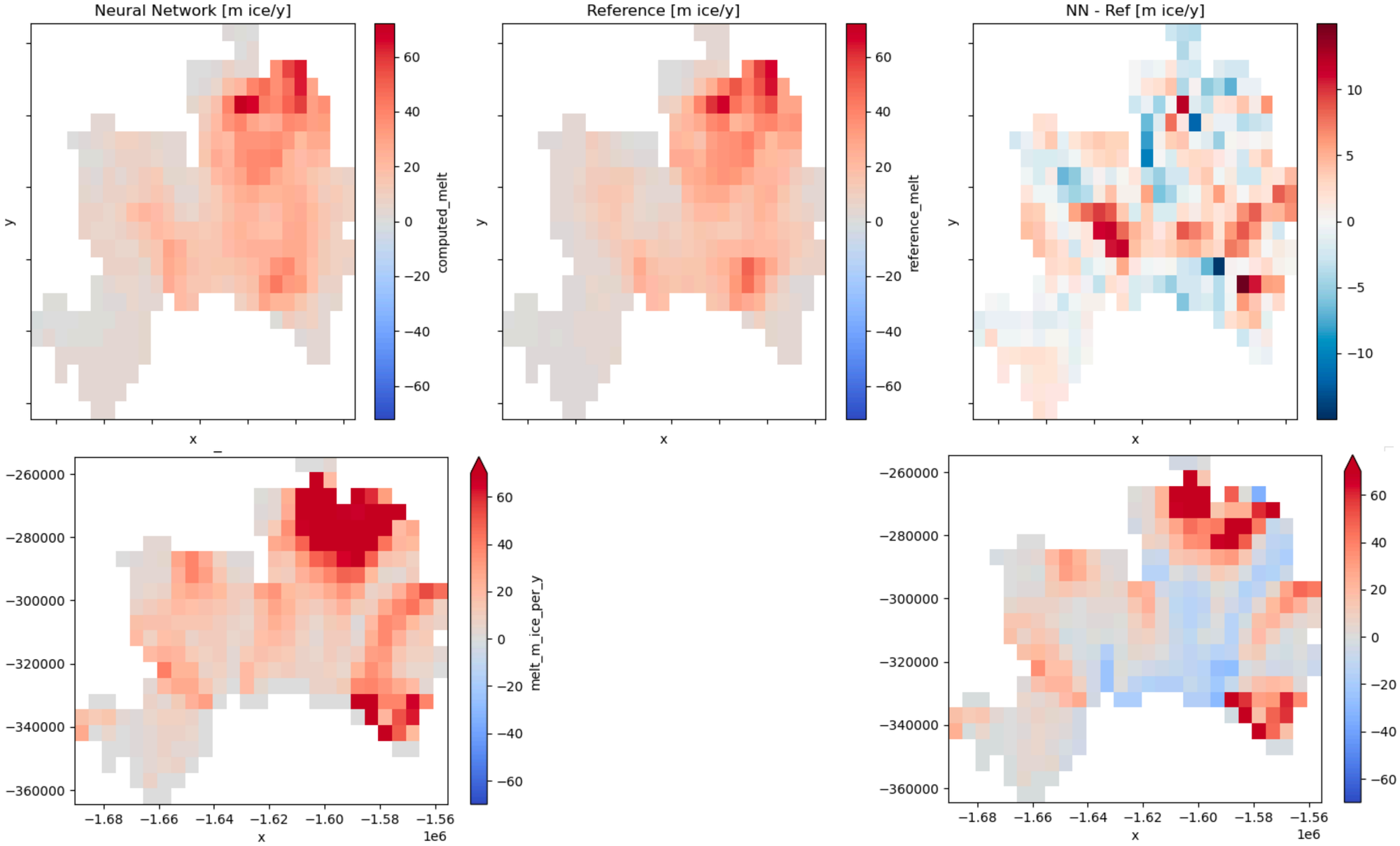
2016



2018

Comparison to existing simple parameterisation

2018



Neural network

$$K \cdot \frac{\text{thermal forcing}}{|\text{thermal forcing}|} \cdot \sin(\text{local slope})$$