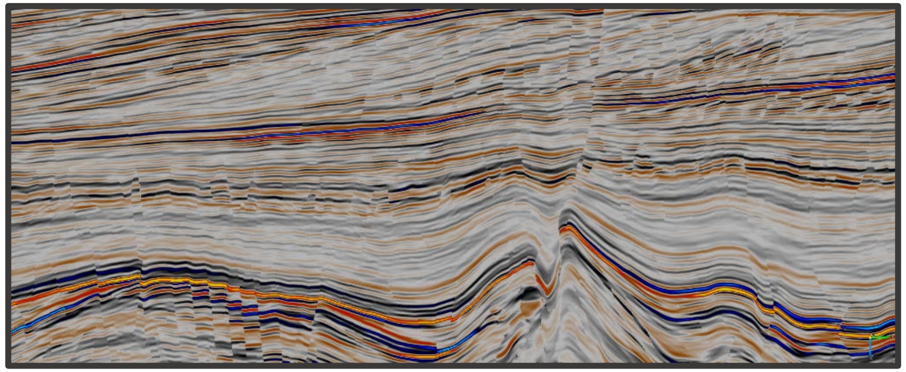
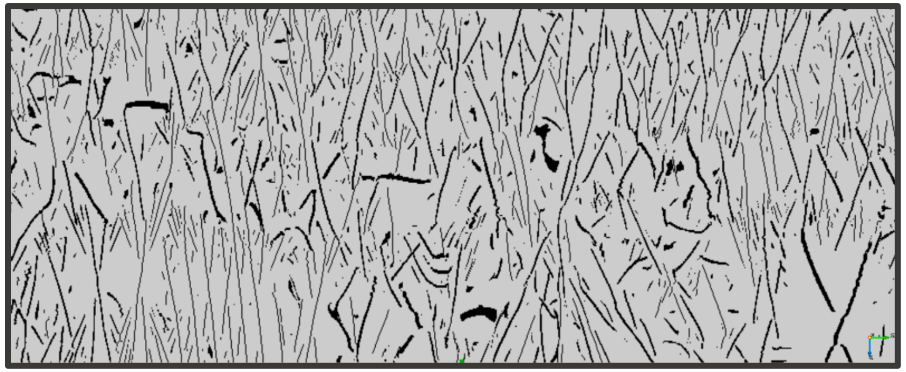
Exercise objective:

To predict seismic features using the *Seismic Image to Image* workflow in the machine learning plugin. In this exercise, we will predict fault locations from seismic data.

Note: To predict real faults use the pre-trained U-Net fault predictor
In this exercise we train a U-Net to predict faults from pre-processed seismic input. The input is Edge-Preserved Smoothed (EPS) seismic data. The target is a mask volume with ones (faults) and zeros (no-faults) that was created from Thinned Fault Likelihood (TFL) computed from the EPS volume. **Note** that from a geoscientific perspective this is not necessary, since we do not need a machine learning model to predict a desired outcome that can be computed directly with an algorithm. The main purpose of this exercise is to learn how to run image-to-image workflows.



Input EPS* seismic



Target mask (0,1) of TFL* from EPS

***EPS and TFL-mask are *NOT* delivered with F3. To replicate this workflow first create EPS and TFL (from EPS) in the Faults & Fractures plugin. Next, create a mask from TFL with the mathematics attribute using this formula: TFL > 0.01 ? 1 : 0**

Exercise objective:

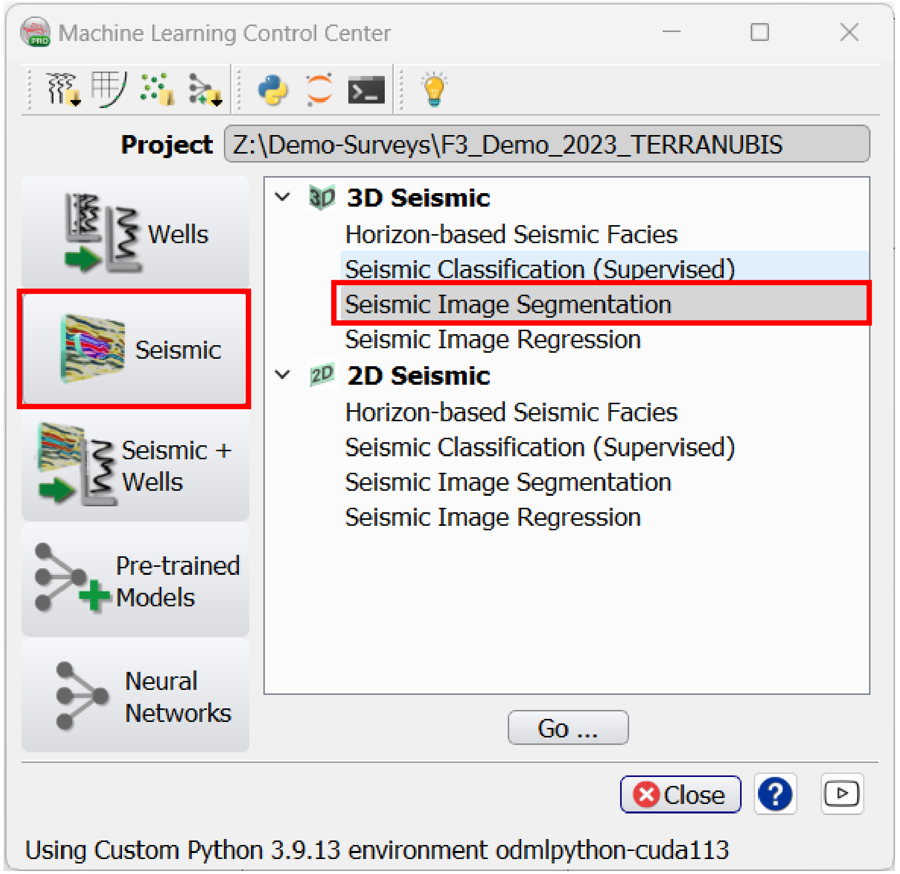
Note: heavy GPU requirements
In this exercise we create 1008 cubelets of 128x128x128 samples. These cubelets are extracted from half the input - and target volumes. The trained U-Net is applied to the full volume. Application is very fast (minutes) but training takes several hours on a GPU. The graphics card used is a Nvidia GeForce with 11 GB DDR6 memory. In principle, the exercise can also be run on a CPU but then training may take several days.

Workflow:

1. **Open** the *Machine Learning Control Center* with the  icon.

2. **Click** on *Seismic*.

3. **Select** *Seismic Image Segmentation* and **Press** *Go*.



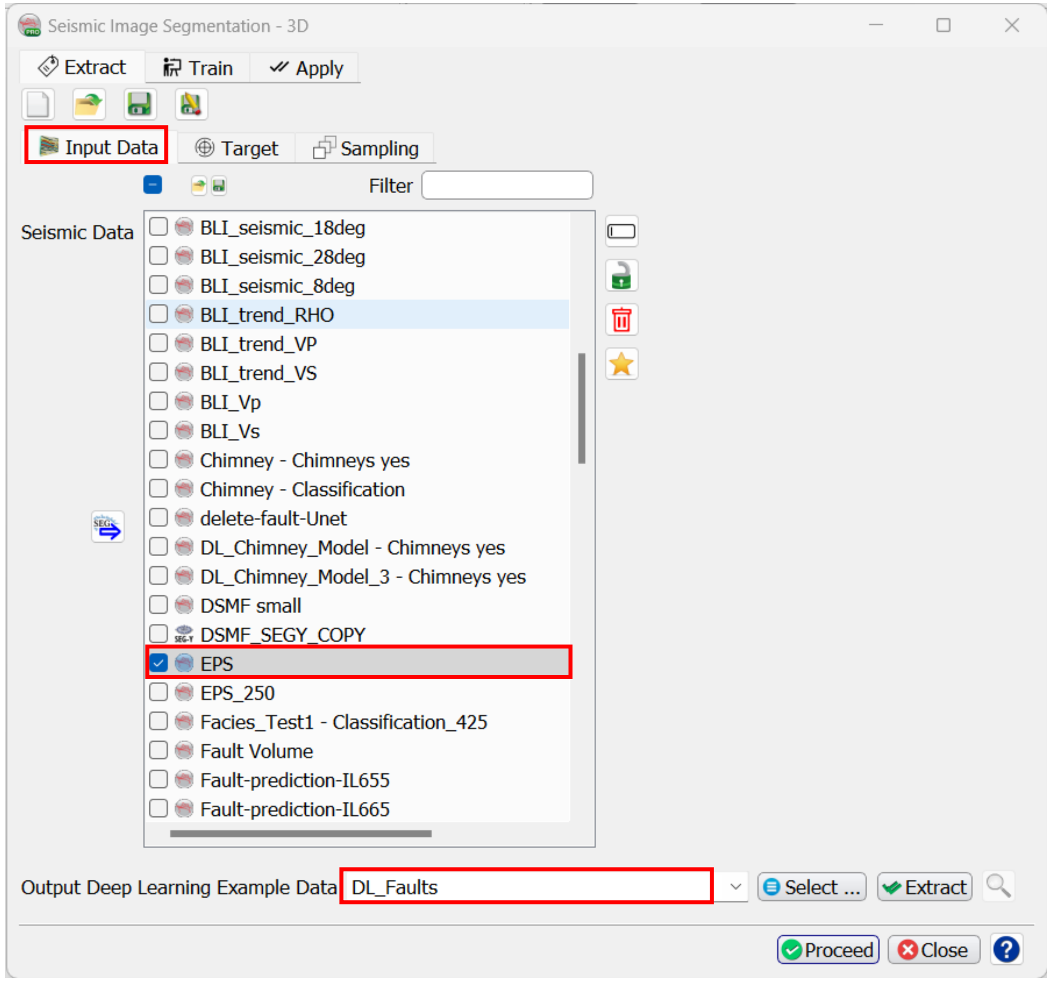
Workflow cont'd:

4. *Seismic Image Transformation* window pops up.

5. **Select** *Input Data* in the *Extract Data* tab.

6. In the *Seismic Data* list, **Select** the *EPS* volume

7. **Specify** a name for the *Output Deep Learning Example Data* and **Press** Proceed.

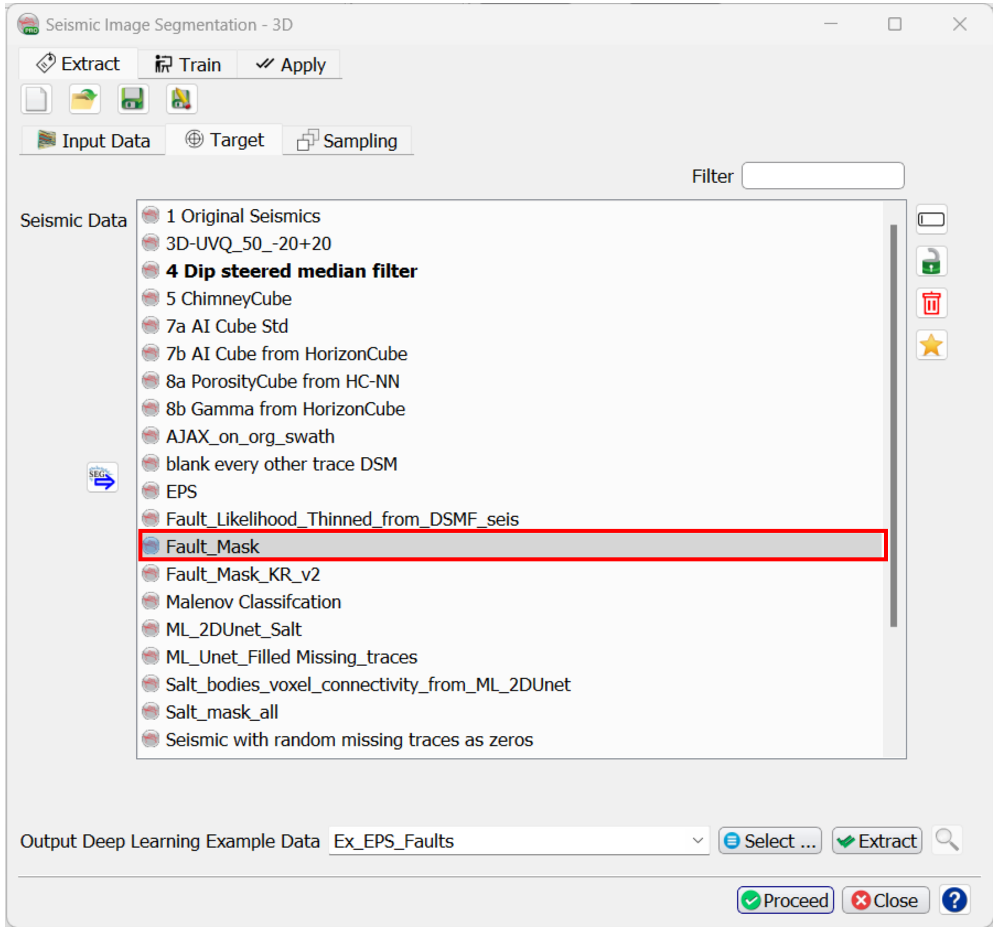


Tip: Additional seismic attributes can be added using checkboxes

Workflow cont'd:

8. The *Deep Learning: Target Seismics Definition* window pops up. Select the *Fault Mask Volume*

9. **Press** *Proceed [Input Data Selection] >>*



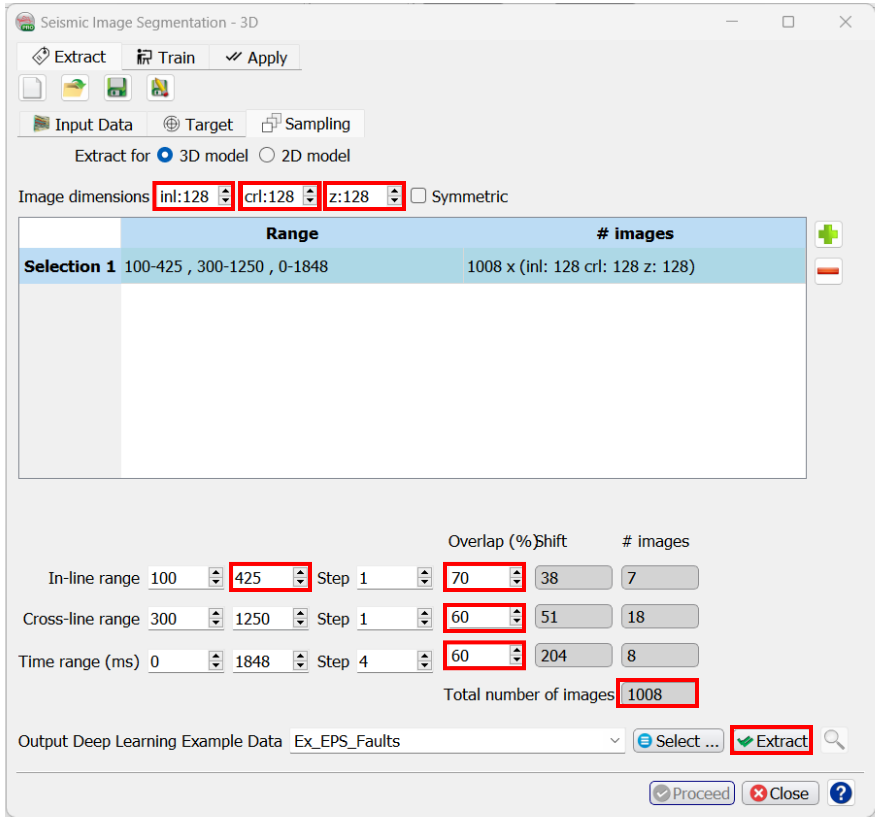
Workflow cont'd:

10. In the *Input Data* window **Set** the *Image dimensions* of the cubelets to 128 x 128 x 128 samples. Note: to extract 2D images, set one of the dimensions to 0.

11. **Specify** the *Inline, Crossline, Time Ranges* and the corresponding *Overlap** percentages to such that we extract approx. 1000 cubelets from one half of the input and target volumes (see image for specifications).

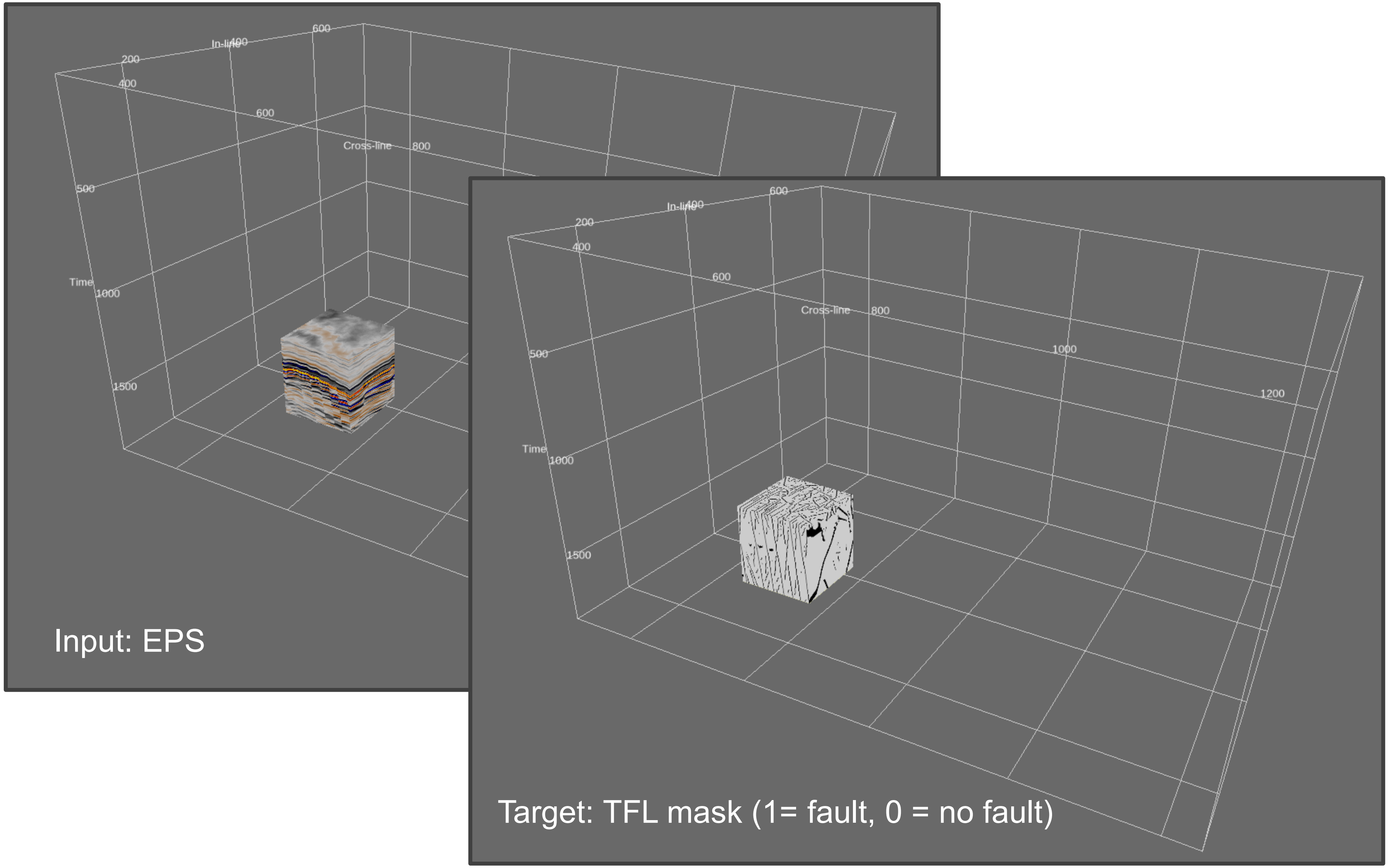
12. **Specify** a name for the *Output Deep Learning Example Data* (e.g. Ex_EPS_Faults) and **Press** Extract

13. When the **Extraction** is done, press Proceed



Overlap: if the number of examples that can be extracted from a given range and overlap does not fit exactly, the last example is extracted from the boundary backwards.

Example cubelets. Dimensions are: 128 x 128 x 128 samples

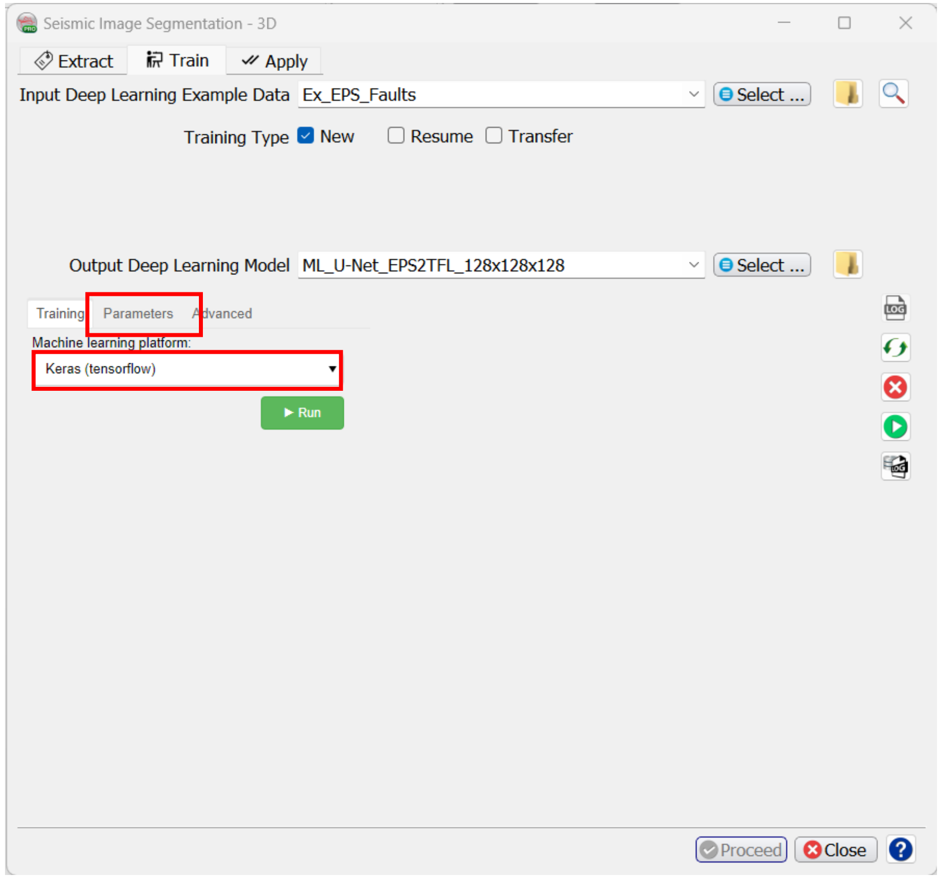


Workflow cont'd:

14. Specify the *Output Deep Learning Model* name (e.g. ML_U-Net_EPS2TFL_128x128x128)

15. In the *Train* tab, **Select** Keras (tensorflow) as *Machine learning platform*

16. **Select** the *Parameters* tab



The machine learning plugin supports two platforms:

Keras (tensorflow) for deep learning (convolutional neural networks) and Pytorch. Supported models and training parameters are specified in the Parameters tab.

Workflow cont'd:

17. In the *Parameters* tab **Select** *Type* U-Net

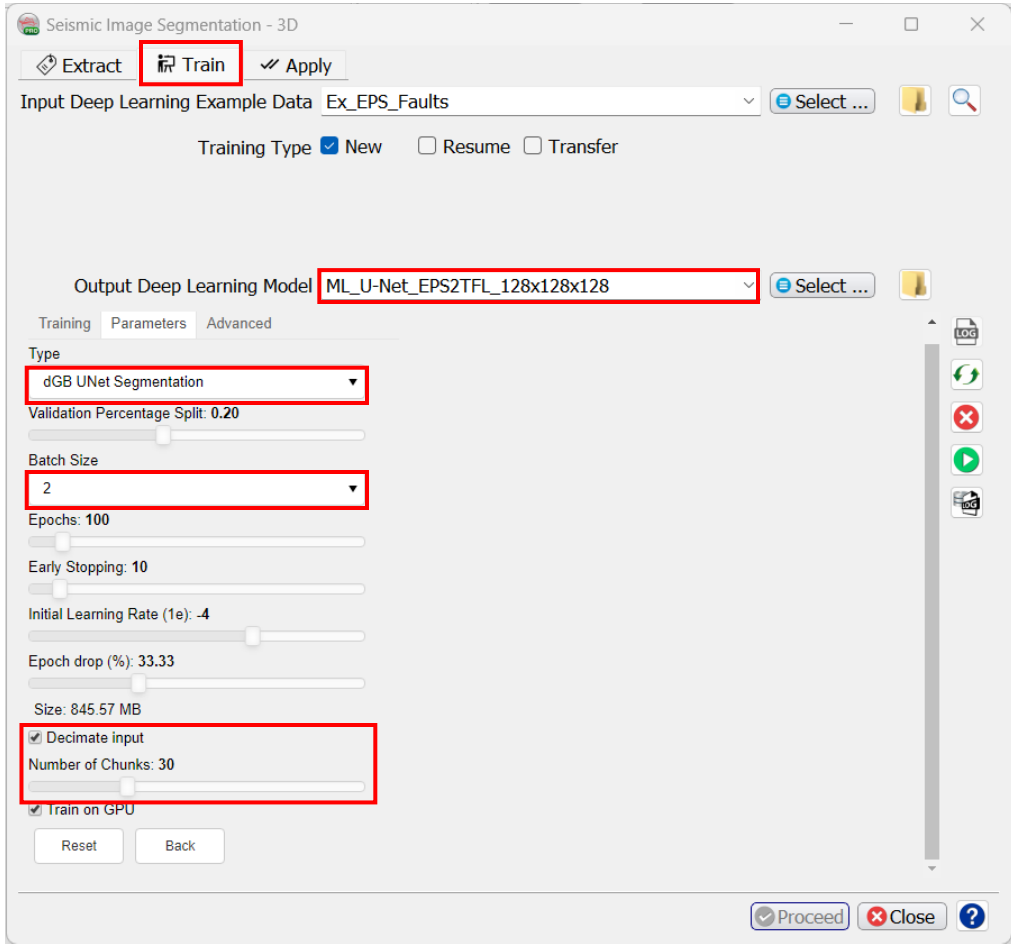
18. Set *Batch Size* to 2. A U-Net needs a lot of GPU memory in the training phase. If memory is exceeded, training stops with an error message. You can then try to rerun with a smaller batch size. Try with the largest possible batch size as training performance increases with batch size.

19. Set the number of *Epochs* to 100 (this is the number of training cycles through all examples that are offered in batches of Batch Size).

20. Set *Early Stopping* to 50. This parameter avoids early stopping when the error does not decrease after this number of Epochs.

21. Go back to the *Training* tab.

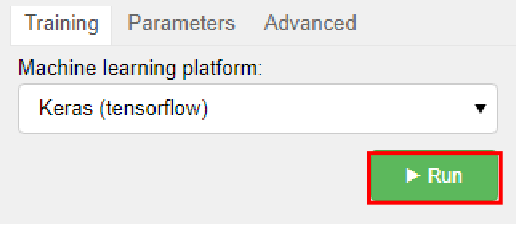
Tip: To change the numbers in the sliders more precisely, click on the corresponding slider and use the arrow keys

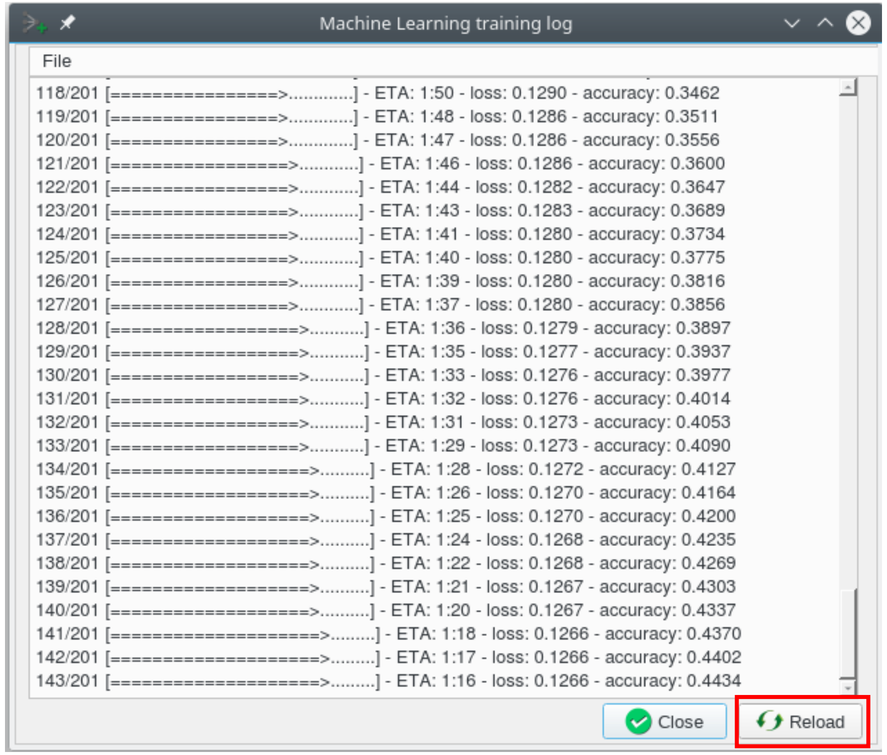


Workflow cont'd:

22. In the *Training* tab **Press** *Run*

23. The Machine Learning training log window pops up. This window can also be started by pressing the LOG icon. **Press** *Reload* to refresh.

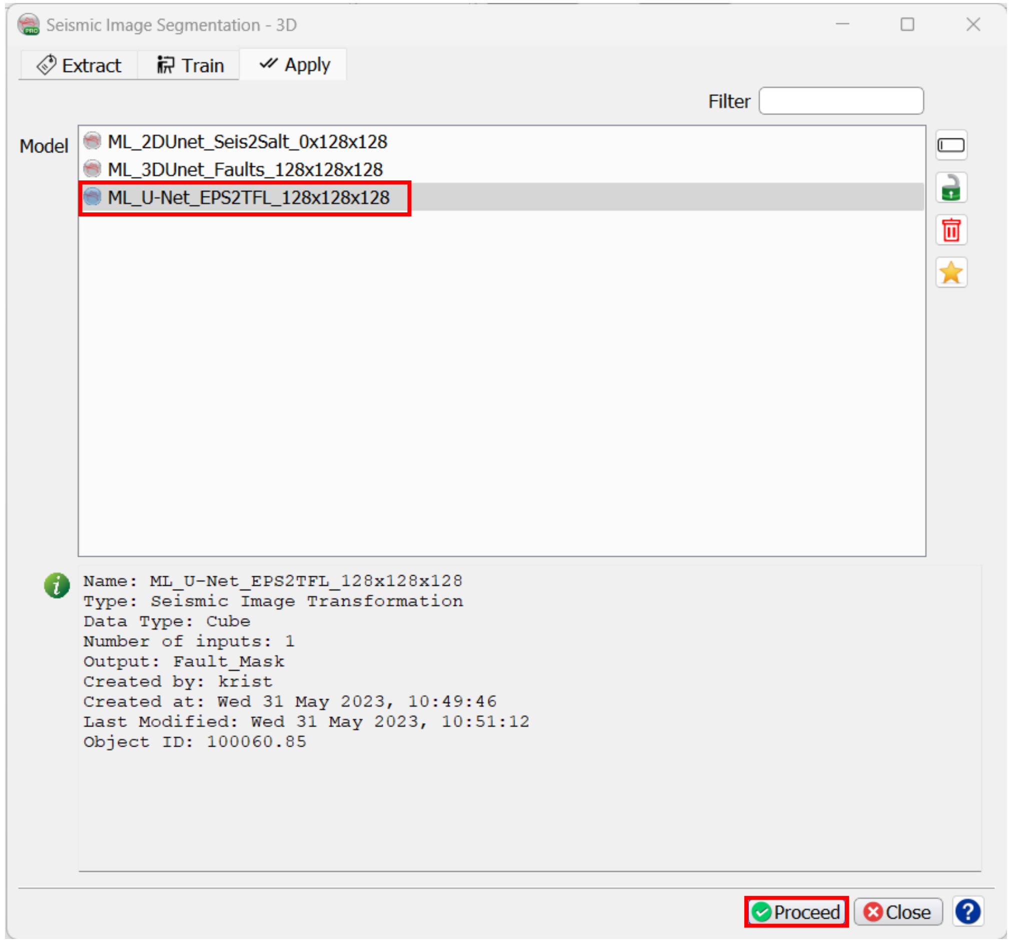




Workflow cont'd:

24. When training is finished **press** Proceed >>
 or **select** the *Apply* tab

25. **Select** the trained model ML_U-Net_EPS2TFL_128x128x128 and **press** Proceed.

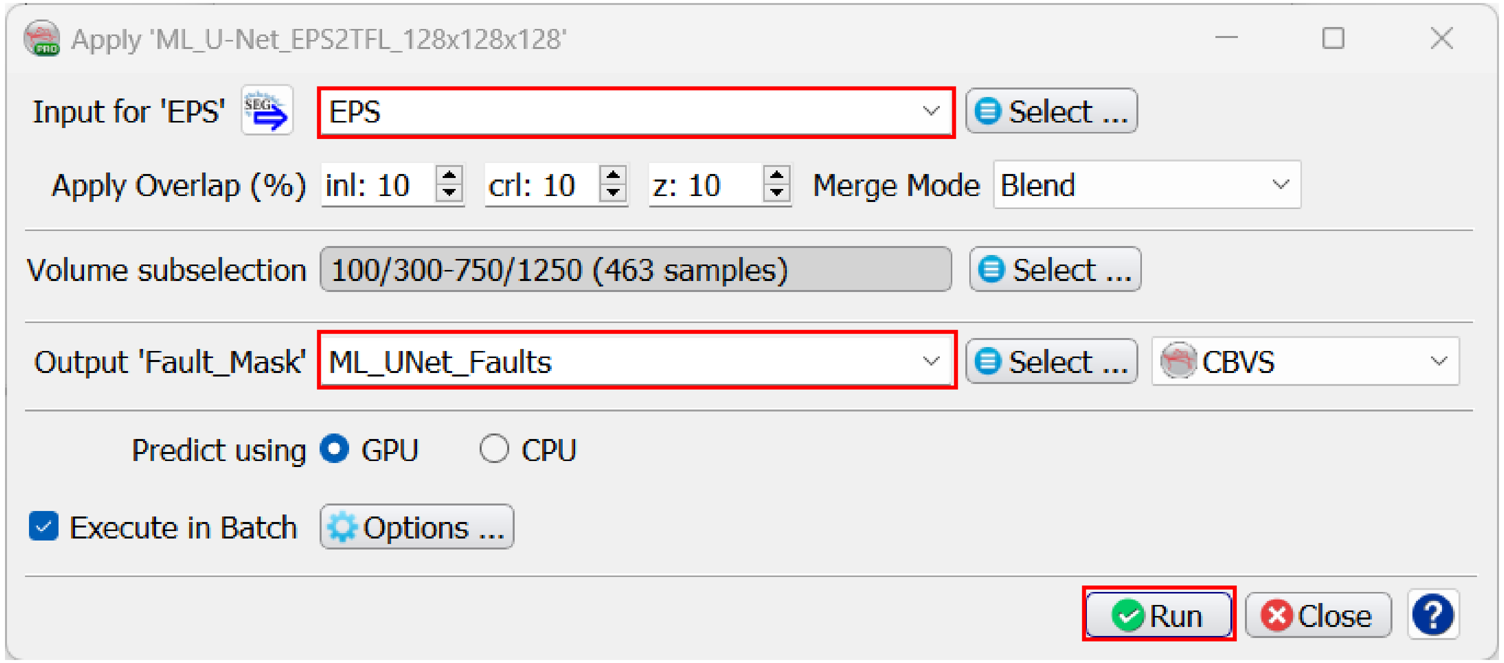


Workflow cont'd:

26. In the *Apply* window **Select** the *Input Cube* Edge_Preserved_Smoothed.

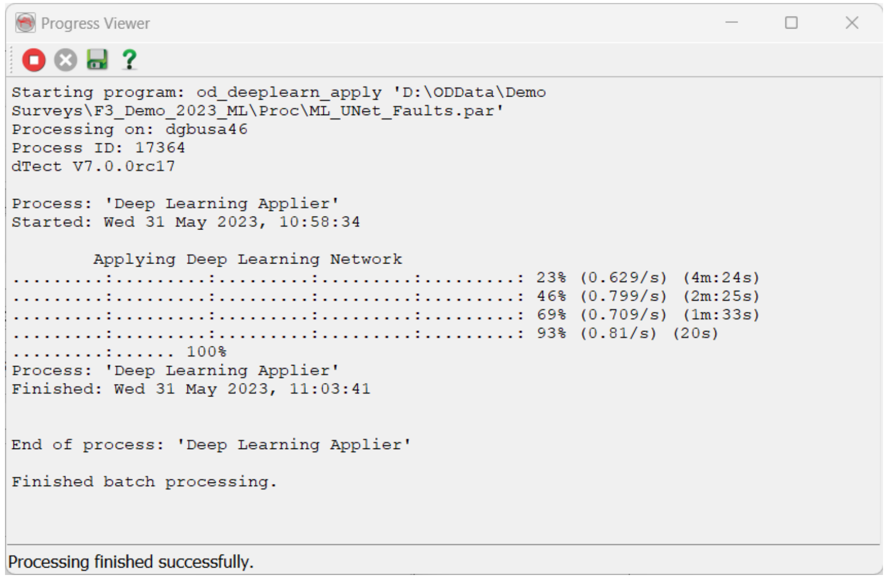
27. Specify the *Output Cube* name that will be created by the trained model, e.g. ML_U-Net_TFL_prediction.

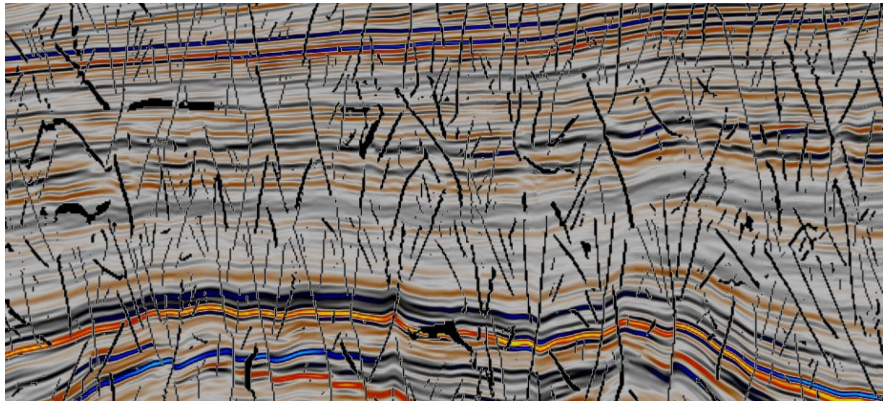
28. **Press** Run to start processing.



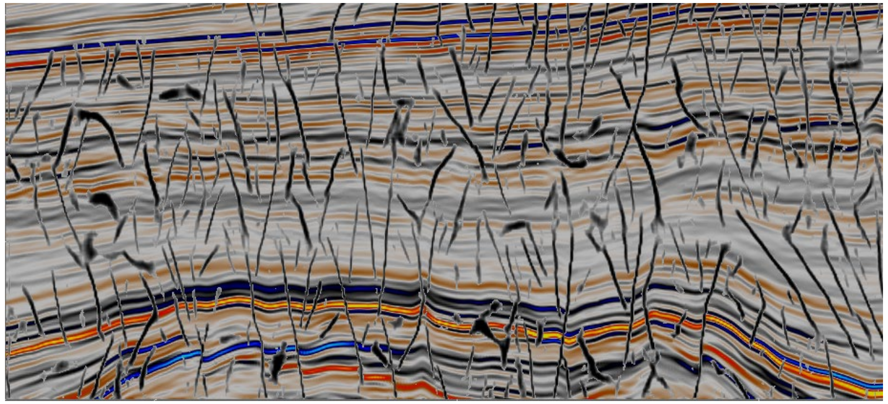
Workflow cont'd:

29. A *Progress Viewer* window pops up. Applying the trained U-Net is very fast. The resulting fault prediction can be viewed e.g. as overlay on the EPS of inline 425.





Inline 500 EPS + TFL mask



Inline 500 EPS + U-Net Prediction