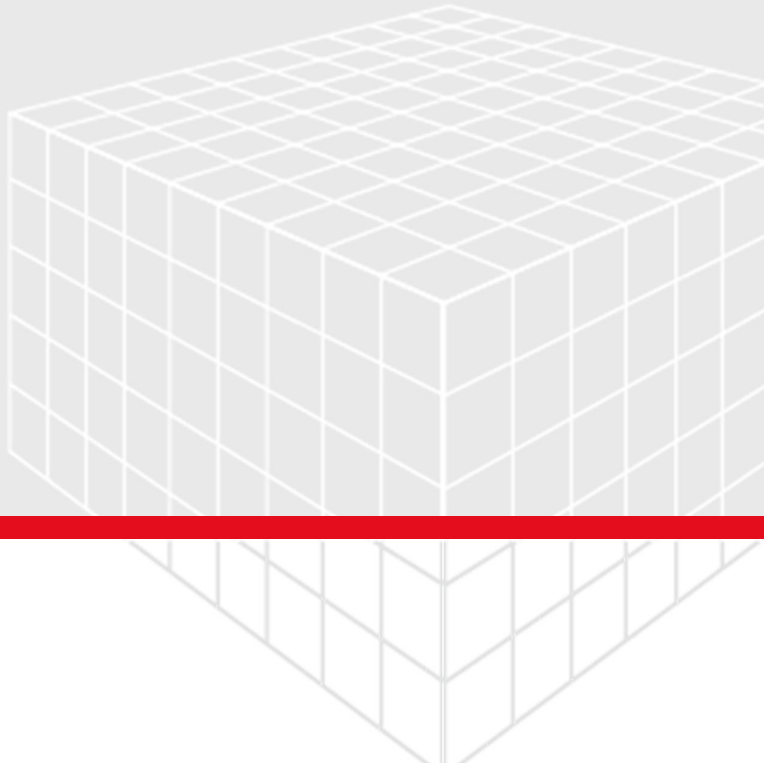
GeoDict-AI

User Guide



GEODICT Release 2026

Published: June 2026

GEODICT

https://doi.org/10.30423/userguide.geodict

GeoDict-AI

1. GeoDict-AI 3

1.1 Theoretical Background 7

1.1.1 Neural Network Training 8

1.2 Preparation of Training Material 15

1.2.1 Generation Script 15

1.2.2 Enhance Grayscale Image 21

1.2.3 Segment Grayscale Image 22

1.2.4 Set-up Training Data Manually 24

1.3 Design of Experiments 27

1.3.1 Options 27

1.3.2 Results 31

1.4 Create Training Data 33

1.4.1 Options 33

1.4.2 Results 35

1.5 Train Neural Network 39

1.5.1 Options 39

1.5.2 Run Training 50

1.5.3 Results 54

1.6 Apply Neural Network 58

1.6.1 Options 58

1.6.2 Results 66

1.7 Enhance Grayscale Image 71

1.7.1 Options 71

1.7.2 Results 75

1.8 Segment Grayscale Image 77

1.8.1 Options 77

1.8.2 Results 83

1.9 Validate Performance 84

1.9.1 Options 84

1.9.2 Results 90

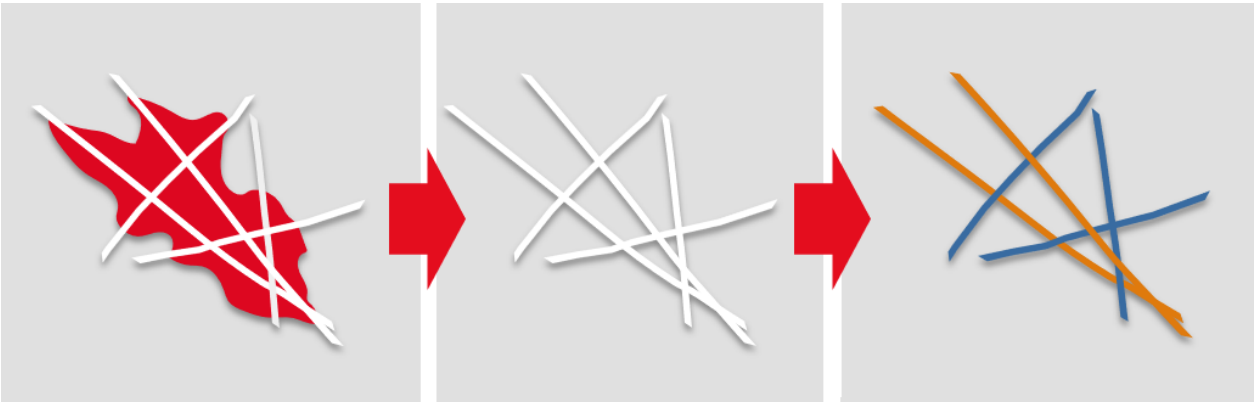
1.10 References 99

1 GeoDict-AI

The GeoDict-AI module covers the functionalities in GeoDict that aim to reconstruct 3D models obtained from segmented computer tomography or FIB/SEM images of different multi-component materials, e.g., nonwovens and electrodes.

The GeoDict-AI module provides the possibility of using **Artificial Intelligence** (AI) approaches for the separation of the different material while image segmentation.

Often, different materials, for example binder and fibers, have the same gray values in the images. Therefore, an automatic separation based on the gray value is not possible. However, it is possible to separate the different materials based on their shape in the image. A neural network can be trained to identify binder in a grain structure, to differentiate two fiber types with different shapes or even to identify the individual fibers. In the graphic it is shown how binder is identified from fibers and fibers from each other.



GeoDict-AI is the starting point to analyze physical properties on the segmented 3D-scans considering the different materials. GeoDict-AI can be used for nearly every structure that can be generated with the GeoDict structure generator modules, e.g., grain structures with binder in GrainGeo or fiber structures with binder in FiberGeo. Of course, you can also use other training material. The training data then must be organized in a defined folder path 24.

Neural networks can be trained for four different application cases. In the following for each of them a recommended workflow is outlined.

Identify individual fibers

To train a neural network to identify individual fibers, the following workflow can be applied:

1. Start with creating a generation script 15 (.py) using the GeoDict structure generator modules. Select the right parameter ranges to match the statistical properties for the considered fibers that should be identified, e.g., fiber diameters, orientations, and the structure density.
2. Then, use the script to design 27 your experiment and to create 33 the training and testing data.

3. Having the training data, train a neural network 39 that can identify the individual fibers in every segmented 3D-scan containing fibers matching the statistical properties caught in the training data.
4. Validate the performance 84 of the trained neural network by applying it to all generated training and testing structures.
5. Finally, use Identify Fibers in FiberFind-AI to identify individual fibers in a segmented 3D-scan.

Distinguish between different materials

For identifying different materials, the following workflow can be applied:

1. Start with creating a generation script 15 (.py) using the GeoDict structure generator modules. Select the right parameter ranges to match the statistical properties for the considered materials that should be distinguished, e.g., the object diameters, the object orientations, and the solid volume percentages.
2. Then, use the script to design 27 your experiment and to create 33 the training and testing data.
3. Having the training data, train a neural network 39 that can identify the target materials in every segmented 3D-scan containing materials matching the statistical properties caught in the training data.
4. Validate the performance 84 of the trained neural network by applying it to all generated training and testing structures.
5. Finally, use Apply Neural Network 58 to apply the trained network to segmented 3D-scans and identify the desired target material.

Enhance grayscale images

Another application of AI is enhancing gray value images. For example, to reduce time and costs in generating CT-scans, a neural network can be trained to enhance the image quality. For example, a scan can be taken from only a few angles or with reduced exposure time.

1. Create low-quality / high-quality image pairs with the same resolution following certain requirements 21.
2. Set-up the training data following a defined folder path 24.
3. Using the training data, a neural network can be trained 39.
4. Apply the network on scans with Enhance Grayscale Image 71.

Segment grayscale images

Train a neural network to Segment Grayscale Image 77.

1. Generate training data.
   a. One possibility is to generate training data using the GeoDict structure generator modules and transform the structures to gray value images using the GeoApp Generate Artificial CT Scan 22.
   b. The other possibility is to use real scans as described here 22.
2. Set-up the training data following a defined folder path 24.
3. Given your training data, a neural network can be trained 39.
4. Apply the neural network with Segment Grayscale Image 77.

New in GeoDict 2026

- Materials can be selected for each class in Apply Neural Network 64.

Learn to use GeoDict-AI:

- Theoretical Background 7
  AI-approach in GeoDict.
- Preparation of Training Material 15
  Learn how to prepare your training material depending on the use case.
- Design of Experiments 27
  Set the design of experiments defining varying parameter ranges and the number of training structures.
- Create Training Data 33
  Create the actual training and testing data with the defined parameter ranges.
- Train Neural Network 39
  Use the training data to train a neural network.
- Apply Neural Network 58
  Apply the trained neural network on segmented scans to identify target materials.
- Enhance Grayscale Image 71
  Reduce time and costs in generating CT-scans by using a neural network trained to enhance the image quality.
- Segment Grayscale Image 77
  Use neural networks trained to segment gray value images.
- Validate Performance 84
  Validate the performance of a trained neural network by applying it to all generated training and testing structures.

Learn to use GeoDict-AI:

- References
  References used in this section of the User Guide.

Join the conversation

Visit the Community on GeoDict Forum to be inspired and get answers to top questions.
Find illustrative application examples on the website.
Our tutorials and seminar videos give you a first hands-on experience. They are collected in the Learning Center.
Send your Support Request to our technical Support.

1.1 Theoretical Background

The fundamental idea of AI is that a neural network is trained to perform a special task, such as the identification of separate fibers, the differentiation between material phases, or to enhance or segment a grayscale image. To train the neural network, enough examples need to be available to teach it. Clearly, for 3D-scans it is a very hard and time-consuming task to manually label materials or objects for the number of examples required. This is where GeoDict's unique structure-generation capabilities come in. The idea is that a neural network trained on these synthetic samples can then be used to analyze 3D scans of real materials. You can use any of the GeoDict structure generators matching your individual application, e.g.:

- FiberGeo can quickly generate a large number of 3D structures of fibrous media. Variation of fiber diameters, shapes, lengths, curvatures and density as well as the amount of binder, if any, can be specified to match the ranges seen in the real materials to be analyzed.
- GrainGeo provides similar possibilities to generate different grain structures.

As long as these structures are close enough to the real 3D-scans, they can be used to train neural networks using GeoDict-AI. For Enhance Grayscale Image, you should provide pairs of registered low-quality / high-quality (CT) scans. The neural network then learns to transform low-quality images to high-quality images. Note however, that even a small number of image pairs (e.g., 2-3) can be sufficient, depending on the application. Depending on the scan, it can also be possible to create such images using the GeoApp Generate Artificial CT-Scan. This app is used in combination with the GeoDict structure generators, generating an image based on a structure file. Thus, this app can also be used to create training data to segment grayscale images.

When applied to a 3D image, the neural network will transform it to another image by assigning a scalar output value between 0 and 1 to each voxel. How this image is interpreted depends on the task at hand:

- When **identifying fibers**, the neural network will mark a centerline which corresponds to a curve along the central axis of each fiber. FiberFind-AI will then automatically reconstruct the individual fiber objects from this output.
- When **identifying material phases** (e.g., binder in a fibrous structure) the different material classes are marked directly by the neural network output.
- For **enhancing grayscale images**, the neural network output corresponds directly to the improved grayscale image.
- When **segmenting grayscale images** a input grayscale image is transformed into a segmented structure file.

For a detailed explanation of neural network training look here 8.

GeoDict-AI uses the Pytorch Framework by The Linux Foundation, one of the most used and well-known machine learning libraries (see also Paszke et. al., 2019 99). The required version of Pytorch is installed during the GeoDict installation and GeoDict-AI works out of the box. Install GeoDict on the used machine, to install Pytorch correctly.

GeoDict-AI may run on the CPU (the main processor) or on the GPU (the graphics card). If one or more suitable GPUs are detected during installation of GeoDict, GeoDict-AI will run on the GPU; otherwise it will run on the CPU.

The GPU version is usually much faster than the CPU version (roughly a factor 10), but it requires a good GPU. For up-to-date recommendations visit our website.

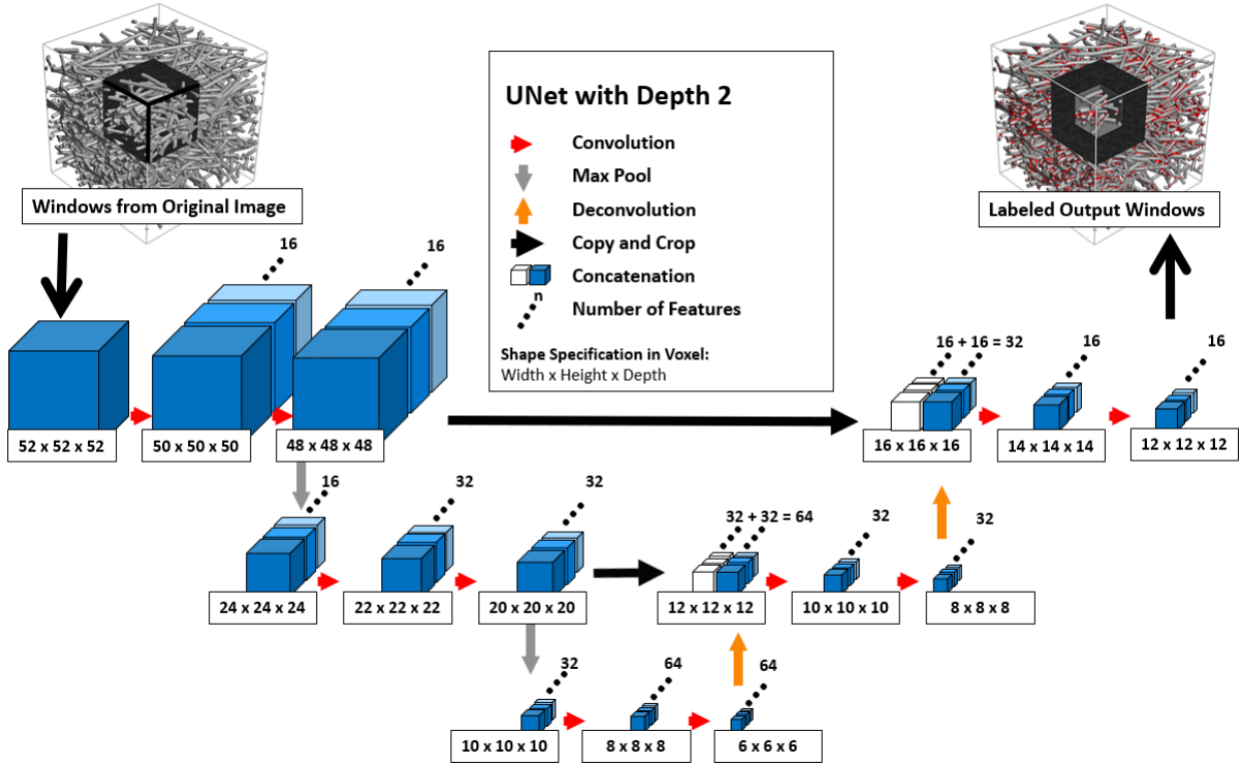
1.1.1 Neural Network Training

To train a neural network, GeoDict-AI uses the **UNet** method Ronneberger and Fischer, 2015 [99], which implements an image-to-image transformation. When illustrated as below, the **UNet** has a U-shape (hence the name) with a convolutional constricting and a deconvolutional expanding branch on the left and right respectively. The constricting branch analyzes and simplifies the input structure to features. The expanding branch uses these features to decide, for example, which of the input voxels to label as binder.

The UNet always works on a fixed input window size and produces results for a smaller output window which is centered within the input window. In order to analyze a given 3D structure, which is usually larger than the input window, the window is automatically shifted over the whole structure and the network is applied at each window location in order to obtain results for the whole domain.

In the diagram, the given size for this input window is 52 voxels in X-, Y- and Z-direction. The window contents are encoded in the left branch of the UNet to an abstract feature map. This feature map is then decoded within the right branch of the UNet to obtain the output image.

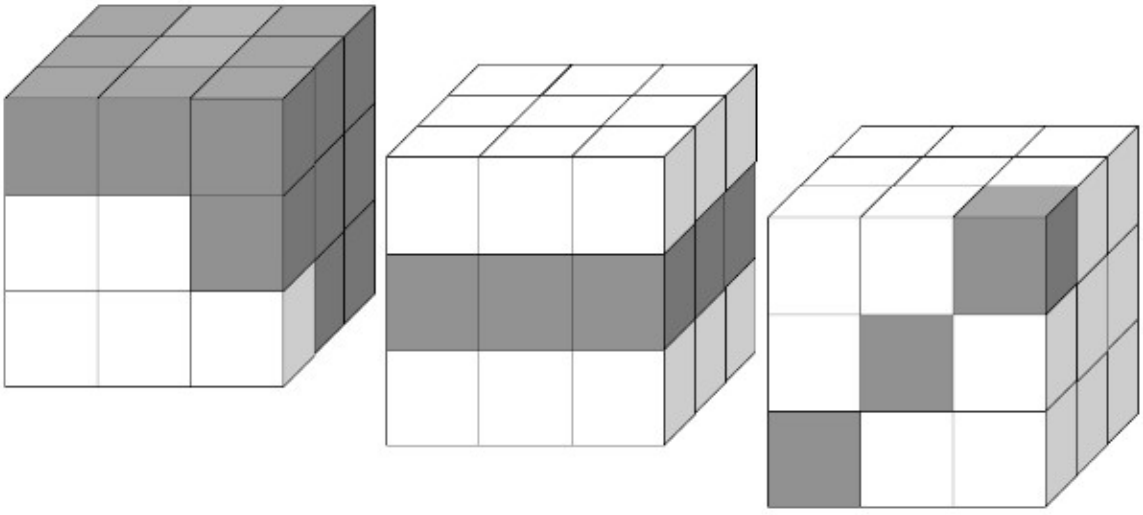
The diagram below shows a UNet with **depth**=2. The depth is defined by the number of max pool operations / deconvolutions leading from **layer** to layer. Thus, a depth of 2 results in three layers.



Operations in one layer of the UNet

In each layer of the UNet two **convolutions** are applied on this cutout, encoding it to discriminative **features**. The first convolution in the first layer of the UNet encodes the input image into a given number of features or feature maps describing the pattern of the cutout. In the diagram above there are 16 features in the first layer.

A convolution analyzes a cutout using **kernels**. A kernel is a feature detector, i.e., a 3x3x3 voxels box with a defined pattern, for example to recognize edges. In the figure below, three exemplary kernels are shown. The features are learned by the network, so that each network produces its own set of features.

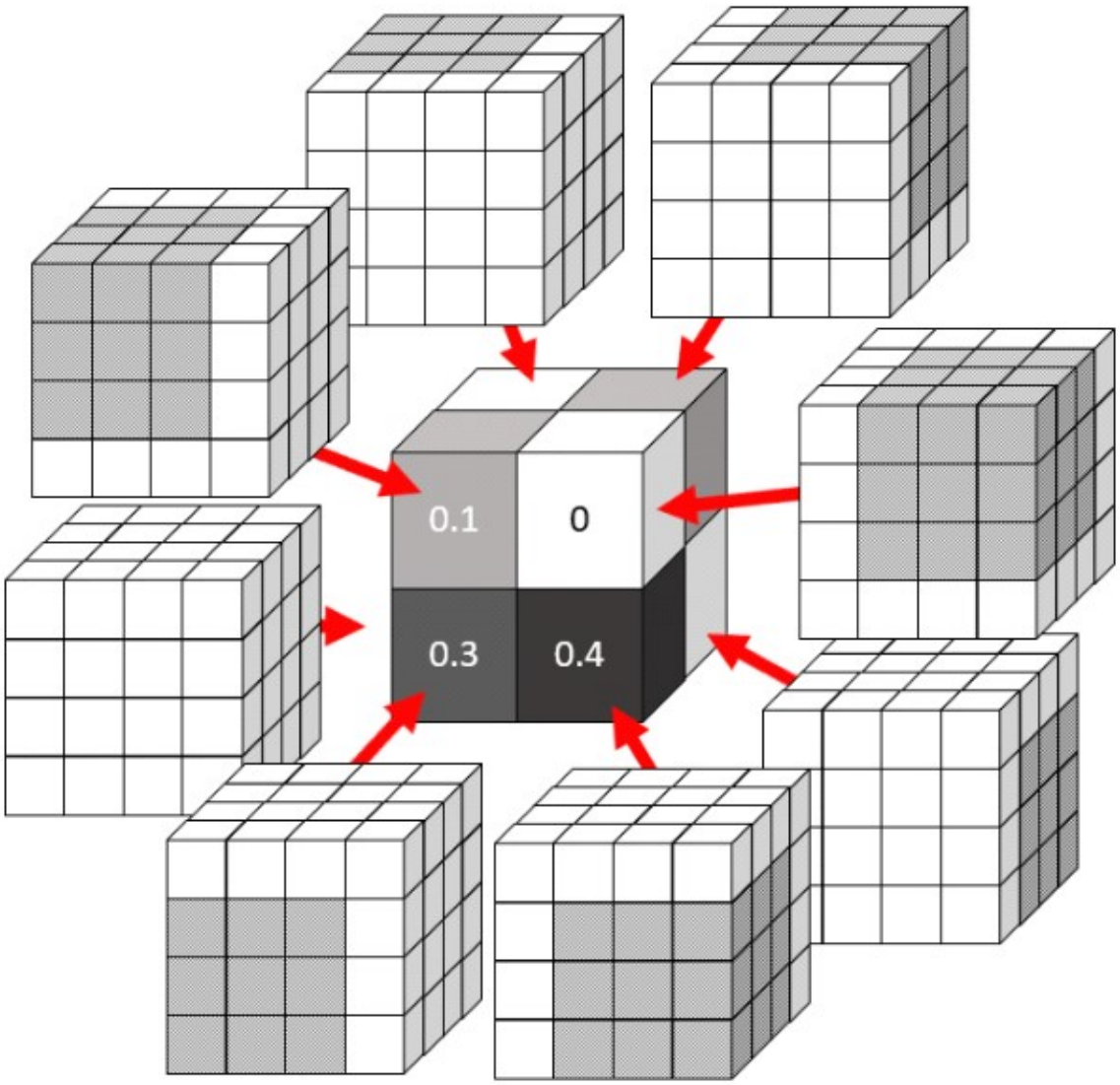


Kernel Examples

For each kernel the **convolution** then decodes the considered structure to one feature map. For this, a kernel scans the cutout voxel for voxel and for each 3x3x3 voxel group, it measures the similarity between the kernel and the underlying image cutout using the dot-product between the kernel values and the image values. The resulting so-called feature map indicates where the feature is found in the input image. For the given network topology, we have 16 features in the first layer, so this process is repeated for each different feature/kernel, resulting in 16 different feature maps.

As the context information from surrounding voxels has to be taken into account for the convolution, the kernel has to stay within the bounds of the input window. Thus, after each convolution the resulting cutout decreases, losing the border voxels. In the UNet diagram above, this results in a cutout size of 48x48x48 voxels after the two convolutions in the first layer.

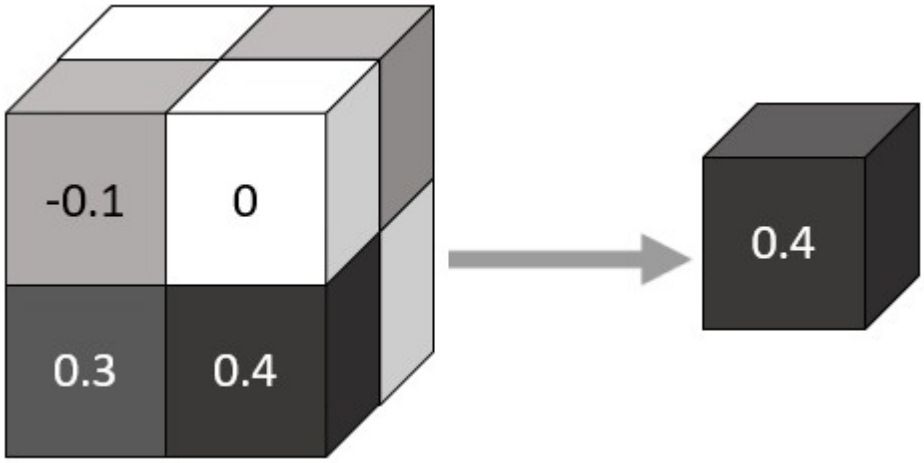
In the figure below, one kernel is applied to a 4x4x4 cutout, resulting in a 2x2x2 feature map where large values correspond to a strong detection of the feature. Regions where the feature is not present will result in low or even negative values in the feature map.



Convolution for 1 kernel

Leaving a layer with Max Pool

The next layer of the UNet is then reached with a **Max Pool** operation. Max pooling halves each spatial dimension by replacing each 2x2x2 block of voxel values by a single voxel containing the maximum value within the block. In our example, this means that the second layer starts with a cutout of 24x24x24 voxels.



Max-Pool Operation

This provides the neural network with a limited form of translational invariance, meaning that it becomes tolerant to slight variations in the location of detected features at the expense of spatial accuracy. We will see later how the UNet is able to compensate for that.

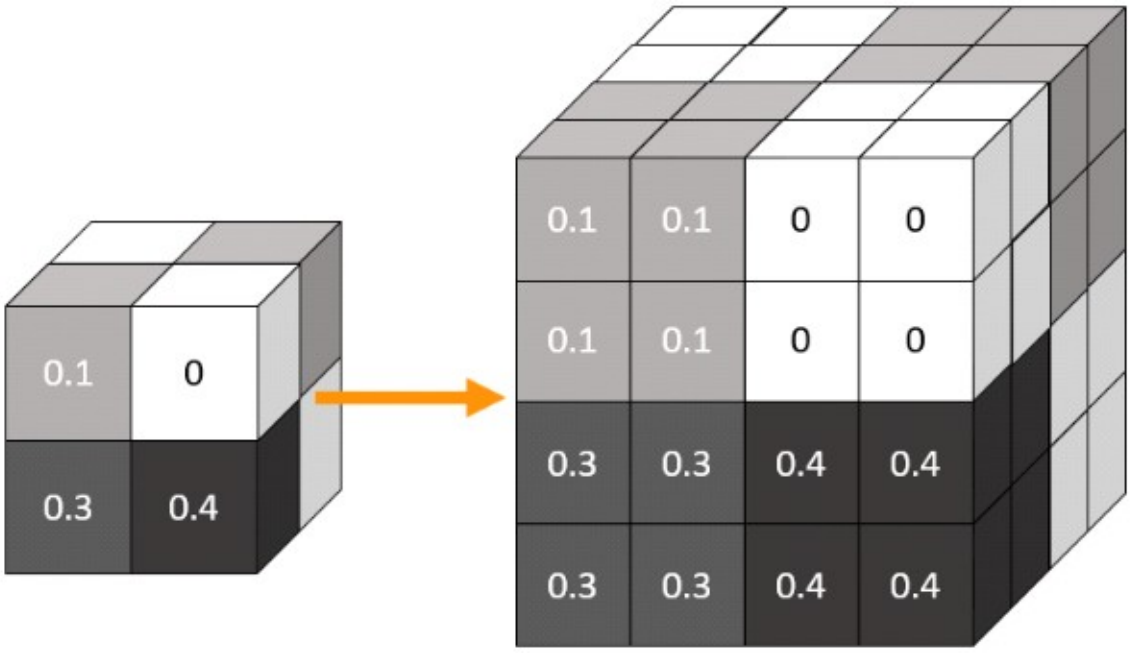
In each new layer of the UNet the first convolution doubles the number of **features**. Thus, the input image is encoded into feature representations at multiple different levels. This is followed by a second convolution with the same number of output features.

End of the last layer

At the end of the last layer, we start moving up the right (expanding) branch of the UNet, which progressively increases the resolution of the image while at the same time reducing the number of features, essentially mirroring the transformation seen in the left (constricting) branch.

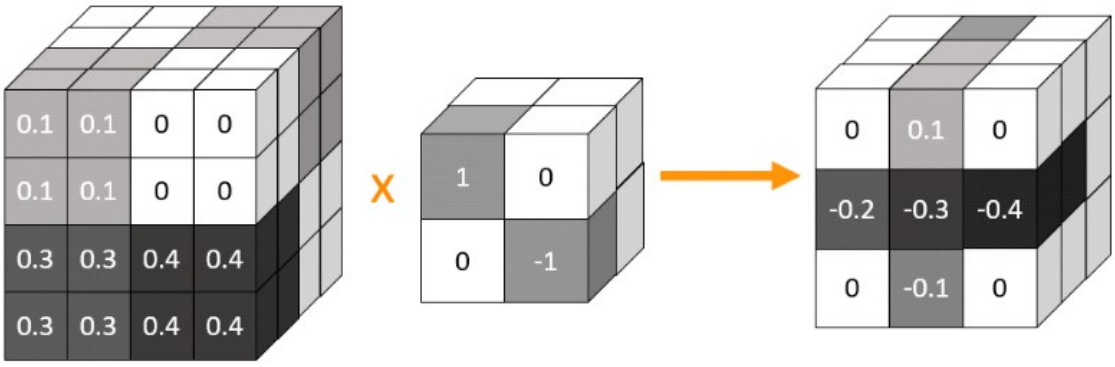
Moving up with deconvolution

As we move up the right branch, the max-pool operation is replaced by a so-called deconvolution (transposed convolution) which doubles the spatial resolution and halves the number of feature maps. In detail, for a deconvolution first an **up-sampling** is applied on the features. For this, each voxel is transformed into 8 voxels with the same value, doubling the dimensions in all three directions.



Up-sampling

Then, a regular **convolution** concretizes the features, by "painting" the voxels in a feature. Here, smaller kernels are used, i.e., 2x2x2 voxels boxes with defined patterns. The feature values then are multiplied with the kernel values as shown in the figure below. Here we show an example with a kernel containing only the values -1 and 1. The kernel is applied for each 2x2x2 voxel group in the input feature map and the dot product is the value for the corresponding position in the new feature.



Convolution of an up-sampled feature map

To **keep the volume dimensions**, in our case, however, zeros are appended to the input feature on three sides before multiplying it with the kernel (padding).

Concatenate features with high resolution cutouts

Afterwards, the resulting features are **concatenated** with high-resolution cutouts from the last features in the left branch in the same layer, doubling the number of feature maps. This way small features with high resolution are combined with low resolution abstract features for context information. This makes it possible to paint the voxels

correctly, since it helps the neurons to build a more precise output based on this information.

Since the cutout in the right branch is still convolved twice in each layer, the final output window is smaller than the input window. In our example with a UNet of depth 2 and a window size of 52x52x52 voxels, the resulting cutout has a size of 12x12x12 voxels. This last cutout, in fact is an image again instead of a feature map.

Weights and Optimizers

At this point we have described the topology of the neural network, but the specific values within the convolutional kernels are not yet specified. These so-called **weights** are determined during the training process using an **optimizer** such as Adam Kingma and Ba, 2016. In our supervised learning setup, we specify pairs of input and desired output images. The optimizer applies the neural network to a given input image, computes the differences between the output and the desired output (the so-called **loss**) and adjusts the weights of the neural network to reduce that error. This process is repeated iteratively until the loss converges to an acceptable value.

1.2 Preparation of Training Material

Preparing good training material is the key for training a well-performing neural network. How to set-up the training data depends on the use case.

When training a neural network to distinguish different materials or identifying fibers in a GeoDict structure file (*.gdt), there are two possibilities to set up training data.

1. GeoDict can generate training data for material and object identification using its powerful structure generators. For this, a generation script must be created as explained here 15. Then this script is used in Design of Experiments 27 and Create Training Data 33 to generate the desired amount of training data.
2. If other training data is available from segmented scans the training data must be set-up in a defined folder path, so that GeoDict recognizes it. How to do that is explained here 24.

If instead a grayscale image should be enhanced, the training material must fulfill certain requirements which are shown here 21. Save the training data in the defined folder path 24.

To train a neural network to segment a grayscale image, the data can be generated with GeoDict as shown here 22. Then, organize the training data as shown here 24.

In all cases, use the created training data in Train Neural Network 39.

Consider the following for the training data creation:

- Generation Script 15
  Create a generation script (.py) for the training data generation with GeoDict-AI.
- Enhance Grayscale Image 21
  The training data for enhancing images must fulfill certain requirements.
- Segment Grayscale Image 22
  Learn how to create segmentation training data using GeoDict.
- Set-up Training Data Manually 24
  Learn how to organize your training data in the expected folder structure for the training.

1.2.1 Generation Script

GeoDict-AI is the starting point to analyze physical properties on the segmented 3D scans, taking into account the different materials. It can be used for nearly every structure that can be generated with the GeoDict structure generator modules, e.g., grain structures with binder in GrainGeo or fiber structures in FiberGeo. To train a neural network, create structure models in GeoDict that have the same statistical properties as the scans to analyze.

For this, write a **Generation Script** and further use it as a mandatory input in the GeoDict-AI section. The **Generation Script** should create a series of structures. This can easily be done in GeoDict because every action is a command that can be saved in a GeoPy macro (*.py). To record a macro, select **Macro - Start Macro Recording** from the menu bar. Then, execute the necessary GeoDict commands (e.g., generate a fiber structure and add binder). Finally, select

Macro - End Macro Recording. The GeoPy Scripting topic page describes how to record and edit macros in GeoDict in detail. Additionally, an example generation script[35] can be found in the installation folder.

To ensure you have representative training data, refer to our checklist[19].

Record script and add variables

For the **Generation Script**, choose a structure generator in GeoDict, e.g., FiberGeo or GrainGeo, and generate a structure matching the scan parameters, while recording it in a macro. Then, add the parameters to vary in the **Variables** section.

The generation script must, at the very least, contain a variable called **gdSeed**. When generating structures for training, the value of gdSeed is automatically varied from one structure to the next. The script must also pass this variable along to the structure generators being used in the script, as described below. Otherwise, all generated structures will be the same.

```
Variables = [
    {
     'Name'            : 'gdSeed',
     'Label'           : 'Random Seed',
     'Type'            : 'int',
    },
  ]
```

The variables section can also have more parameters. In the example, a neural network is trained to identify binder in a fiber structure with varying solid volume percentage of fibers and binder. Therefore, variables are added for solid volume percentage and binder solid volume percentage in the Variables section.

Here, only variable 2 (gd_SVP) uses all possible keys. For example, the built-in default value can be omitted, as it is done for variables 1 (gdSeed) and 3 (gd_BinderSVP). This value is not needed for the training. It is only shown in the Macro Execution Control, when executing a parametric macro as described in the GeoPy Scripting chapter.

```
Variables = [
    {
     'Name'            : 'gdSeed',
     'Label'           : 'Random Seed',
     'Type'            : 'int',
    },
    {
     'Name'            : 'gd_SVP',
     'Label'           : 'Solid Volume Percentage',
     'Type'            : 'double',
     'Unit'            : '%',
     'ToolTip'         : 'Solid volume percentage of the created structure.',
     'BuiltinDefault' : 10.0,
     'Check'           : 'min20;max80'
    },
    {
     'Name'            : 'gd_BinderSVP',
     'Label'           : 'Binder SVP',
     'Type'            : 'double',
     'Unit'            : '%',
     'Check'           : 'min2;max15'
    }
   ]
```

The parameters defined in the Variables section then need to be inserted at the right places in the corresponding dictionaries.

The variable gdSeed is entered as a value for RandomSeed, which is a parameter of most structure generators in GeoDict.

```
Create_args_1 = {
   'ResultFileName'           : 'FiberGeo.gdr',
   'MaterialMode'             : 'Material',      # Possible values: Material,
   'MaterialIDMode'           : 'MaterialIDPerObjectType',# Possible values:
   'Parallelization' : {
   'Domain' : {
   'MaximalTime'              : (24, 'h'),
   'OverlapMode'              : 'RemoveOverlap', # Possible values: AllowOver
   'NumberOfObjects'          : 100,
   'StoppingCriterion'        : 'SolidVolumePercentage',# Possible values: Sc
   'SolidVolumePercentage'    : (gd_SVP, '%'),
   'Grammage'                 : (10, 'g/m^2'),
   'SaveGadStep'              : 10,
   'RecordIntermediateResult' : False,
   'InExisting'               : False,
   'KeepStructure'            : False,
   'WeightPercentage'         : (0, '%')
   'Density'                  : (0, 'g/c
   'RemoveOverlap' : {
   'MatchSVFDistribution' : {
   'PercentageType'           : 0,
   'RandomSeed'               : gdSeed,
   'IsolationDistance'        : (0, 'm')
   'MatrixDensity'            : (0, 'g/c
   'OverlapMaterial' : {
   'ContactMaterial' : {
   'NumberOfGenerators'       : 2,
   'Generator1' : {
   'Generator2' : {
   'Temperature'              : (293.15,
   }
gd.runCmd("FiberGeo:Create", Create_arg
```

```
AddBinder_args_1 = {
   'ResultFileName'           : 'Binder.gdr',
   'MaterialMode'             : 'Material',          # Possible values: Material
   'BinderMaterial' : {
   'ContactAngle'             : (0, 'Deg'),
   'StoppingCriterion'        : 'SolidVolumePercentage',# Possible values: Solic
   'SolidVolumePercentage'    : (gd_BinderSVP, '%'),
   'Grammage'                 : (10, 'g/m^2'),
   'WeightPercentage'         : (20, '%'),
   'MaterialDensity'          : (1, 'g/cm^3'),
   'BinderDensity'            : (2.70009, 'g/cm^3'),
   'PeriodicX'                : False,
   'PeriodicY'                : False,
   'PeriodicZ'                : False,
   'BinderAnisotropy'         : 1,
   'DistributionMode'         : 'Homogeneous',       # Possible values: Homogene
   'LayerDensities'           : [0, 0.25, 0.5, 0.75, 1, 1.25, 1.5, 1.75, 2],
   'Temperature'              : (293.15, 'K'),
   'SVPCorrection'            : False,
   }
gd.runCmd("FiberGeo:AddBinder", AddBinder_args_1, Header['Release'])
```

The variable **gdSeed** is used to set up the random number generation for the structure generator. Additionally, if **Random** was selected for **Variation** in the Design of Experiments 29 dialog, gdSeed is used to set up the random number generation for the other parameters defined in the Variables section.

Define the script's output

The requirements for the output of the Generation Script vary depending on the selected postprocessing mode. For fiber identification, at the end of the script there should be a fiber structure in memory, including the analytic object representation of the fibers (GAD data). This requirement is fulfilled for the previous example which generates a fiber structure. Adding binder material also keeps the GAD information about the fibers intact. Note that not all operations will preserve GAD data. To learn more about GAD objects, refer to the GadGeo chapter.

When distinguishing Material Phases[34] instead, the generation script must explicitly write a pair of *.gdt files (GeoDict structure files) as follows:

- **input.gdt**: In this structure all solid material phases must be assigned to material ID 01.
- **output.gdt**: In this structure the different solid materials must all have their own material ID, for example ID 01 for circular fibers, ID 02 for cellulose fibers and ID 03 for binder. For this, reassign all solid material IDs, that are NOT the target materials, for example remaining overlap. Also, objects of the same type should have the same ID. For example all circular fibers should have the same ID.

Consider a **Generation Script** which produces a fiber structure with two types of circular fibers and binder where material ID 00 is pore space, material IDs 01 and 02 are fibers, and material ID 03 is the added binder material phase. To complete the script for GeoDict-AI, the correct material IDs must be assigned and the two output structures saved as follows:

1. If the two fiber types don't need to be distinguished, assign material ID 02 to material ID 01 with ProcessGeo. This option can be found when selecting **Model - ProcessGeo** from the menu bar and selecting **Reassign Materials & IDs** from the first and **Reassign Material ID** from the second pull-down menu in the ProcessGeo section. When recording the command in a macro, it looks as follows:

```
Reassign_args_1 = {
   'AssignMaterial' : False,
   'SwapIDs'        : False,
   'OldMaterialID'  : 2,
   'NewMaterialID'  : 1,
   }
gd.runCmd("ProcessGeo:Reassign", Reassign_args_1, Header['Release'])
```

2. If the structure contains overlap, reassign the overlap to the material ID of one of the fiber types.
3. It is recommended to use consecutive IDs for the different materials, i.e., in this case reassign the binder to ID 02. The creation of the training material and the training of the neural network will also work fine, if the IDs are not consecutive. However. when applying the network[67] you will get empty result fields for the "missing" IDs and for Validate Performance[90] you will obtain nan (not a number) entries under the Report tab for the missing IDs. This does not influence the result quality.

4. Then, save the resulting structure as output.gdt (**File - Save Structure as...**). The key *FileVersion* defines the file format and the value 3 means *.gdt, which is the default structure file format since GeoDict 2023. Thus, the command will look as shown above when it is recorded in a macro.

```
SaveFile_args_1 = {
  'FileName'    : 'output.gdt',
  'FileVersion' : 3,
  }
gd.runCmd("GeoDict:SaveFile", SaveFile_args_1, Header['Release'])
```

5. For the input.gdt file reassign all solid materials also to material ID 01 to obtain a structure, with only pore space and solid material. Then save the structure as input.gdt (**File - Save Structure as...**).

```
SaveFile_args_1 = {
  'FileName'    : 'input.gdt',
  'FileVersion' : 3,
  }
gd.runCmd("GeoDict:SaveFile", SaveFile_args_1, Header['Release'])
```

All these commands do not need to be typed in the text editor, but can be recorded directly from GeoDict as described in the GeoPy Scripting topic.

Especially for structures containing binder it can be necessary to generate the structures bigger than intended and then crop them with **Model - ProcessGeo - Crop** to prevent boundary effects before saving the structures to output.gdt and input.gdt.

Checklist for Structure Parameters

The parameters for structure generation must be chosen to match the properties of the target scan. Ensure to cover the complete required parameter space, by capturing all variations observed in the target materials.

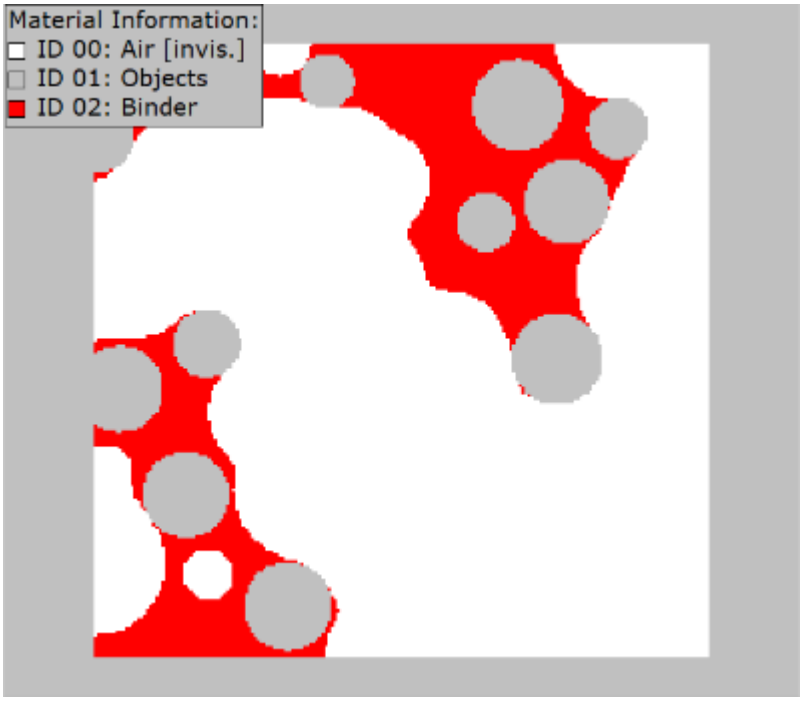
In the following a parameter checklist is given:

- Voxel length (resolution)
- Solid volume fractions for the different materials
- Object diameter ranges
- Object shapes
- Object orientation distribution
- Object overlap (set overlap not larger as is necessary to represent artifacts in the scan)
- Fiber curvature
- CT artifacts (e. g. hour glassing and fiber crossings)
- Contact angle of binder
- Binder anisotropy
- ...

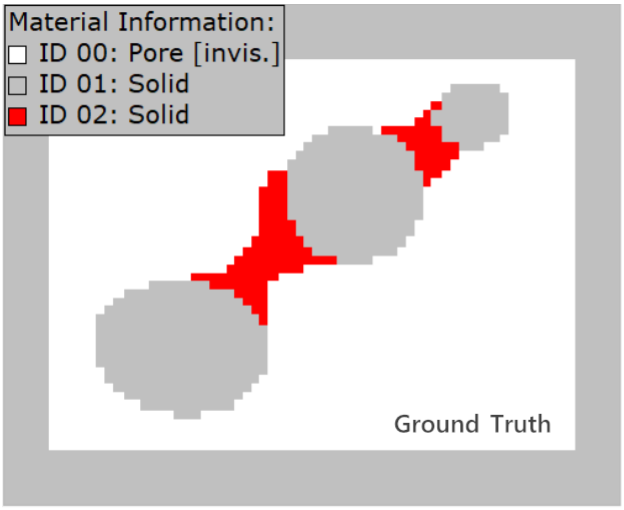
The numbers of voxels in the X-, Y- and Z-direction define the **Domain Size** and should already consider the Window Sizes[43] used for training. For example, for the default window size of 52^3 voxels, a domain size between 256^3 and 512^3 voxels is recommended. Larger structures require higher hardware resources.

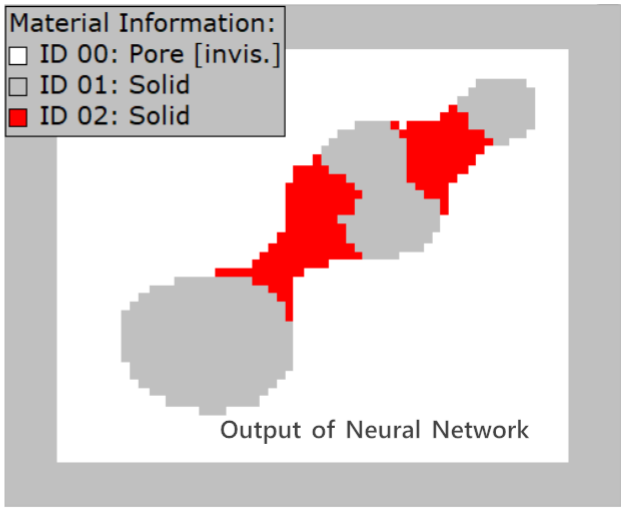
Make sure to **check on boundary effects** when generating structures. Cropping the structure with **ProcessGeo** → **Crop** may be necessary in some cases to avoid them.

Usually, if the different material phases or the individual fibers can be **recognized by eye**, the neural network can also learn to distinguish them. Otherwise, if for example the binder solid volume percentage is very high compared to the object diameters, the objects can "disappear" inside the binder. Then, it is nearly impossible for the network to identify the small objects inside the binder.

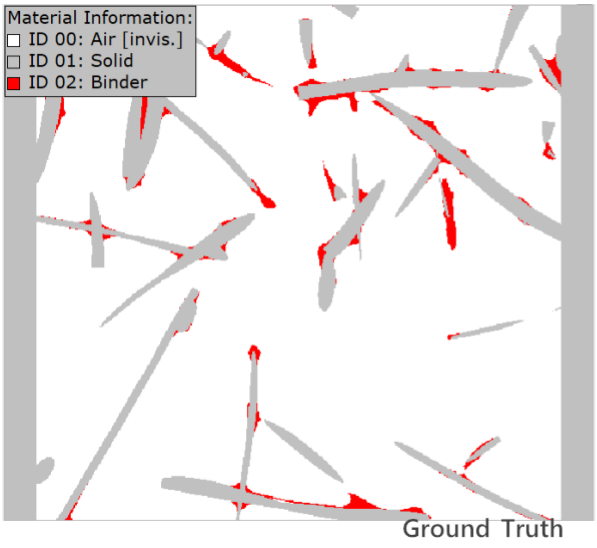


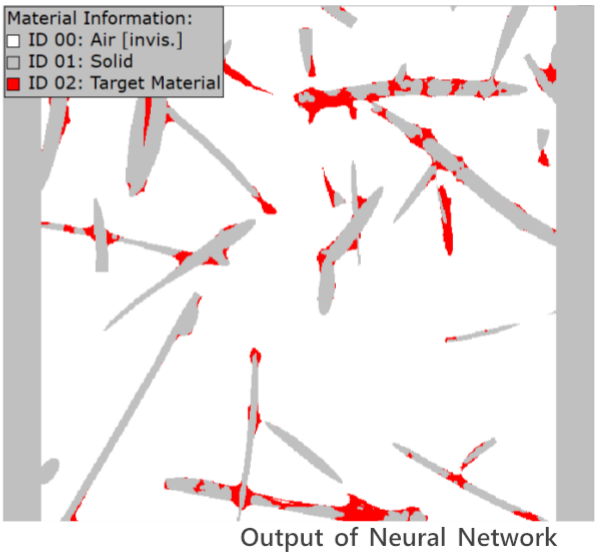
Avoid large parameter ranges, since the training of the neural network would become very difficult. Consider for example a fiber structure with binder. If there are fibers with very different diameters, probably many of the thicker fibers will be recognized as two thinner fibers with binder in between and the other way around.





Make sure to **remove the overlap**, only leaving the amount of overlap that is necessary to represent artifacts in the scan.





1.2.2 Enhance Grayscale Image

While ImportGeo-Vol has many effective image filters, there are cases, where a neural network performs better. Use GeoDict-AI to train a neural network to enhance grayscale images, for example for CT-scans taken from only a few angles or with reduced exposure time. The trained neural network can either be used with Enhance Grayscale Image 71 or in the ImportGeo-Vol Image Processing dialog.

To train the neural network, you need low-quality / high-quality image pairs.

These must fulfill the following requirements:

- they must be registered, i.e., capture the same region,
- currently, they must have the same resolution and therefore, the same number of voxels in X-, Y- and Z-direction,

- the low-quality scan should be saved as input.grw and the high-quality scan should be named output.grw.

The neural network will then learn to transform low-quality scans to high-quality scans such that ideally you will be able to obtain comparable high-quality results by applying the trained network to future low-quality scans. For this, the files must be organized in the folder structure described here 24.

1.2.3 Segment Grayscale Image

ImportGeo-Vol already has a powerful AI-segmentation feature, where you can label a grayscale image to train a neural network to segment the image.

Since GeoDict 2025 it is also possible to train a neural network to Segment Grayscale Image 77 using the GeoDict-AI module. Create training data that match the statistical properties of the scans you want to segment with the trained neural network. Given your training data, a neural network can be trained 39 and applied with Segment Grayscale Image 77. After producing training data pairs each consisting of one gray value image (**input.grw**) and one segmented structure (**output.gdt**), organize the files in the folder structure described here 24.

There are two possibilities to generate the training data, explained in the following.

Create training data from real scans

Select representative scans having the same number of materials IDs after segmenting and needing the same image processing steps. For each of these scans do the following to create the training data:

1. Import the scan in ImportGeo-Vol.
2. Save the scan as it is as **input.grw**.
3. Apply image filters as needed.
4. Segment the scan for example using AI-segmentation.
5. Post-process the resulting structure to remove artifacts for example with ProcessGeo-Cleanse.
6. Save the resulting structure as **output.gdt**.

After all this effort you train a neural network that allows you to skip the time-consuming image processing steps for future scans to segment.

Create artificial training data

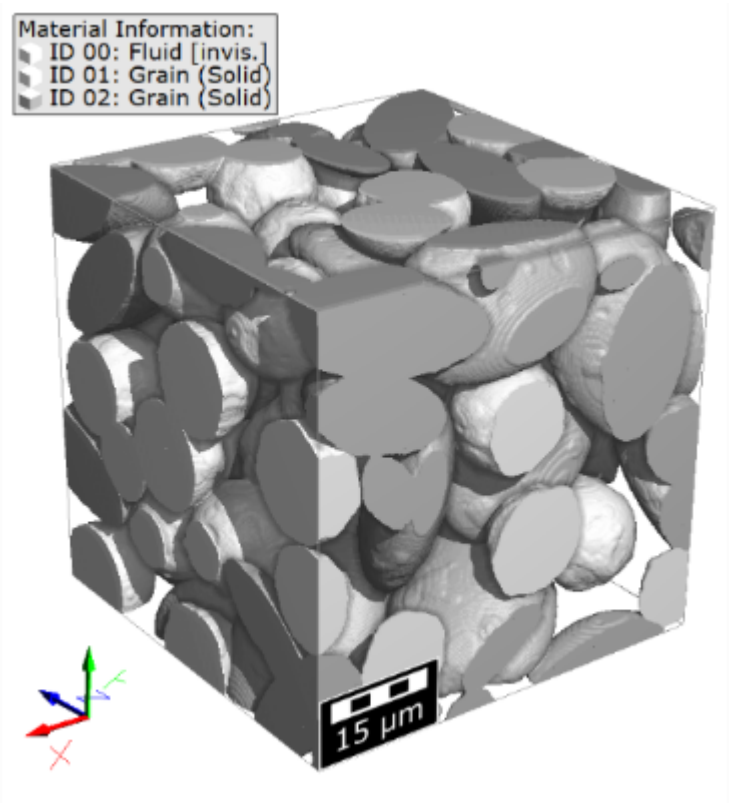
Use the GeoDict structure generators and transform the structures to gray value images using the GeoApp **Generate Artificial CT-Scan**. This app can add noise and many other kinds of image artifacts to the image, generating an image as close as possible to real gray value images. The created structure models in GeoDict should have the same statistical properties as

the scans to segment. To ensure having representative training data, have a look at our checklist 19.

In the following it is described shortly how the User Guide example was created with GrainGeo and the Generate Artificial CT-Scan GeoApp.

Model the structure with GeoDict

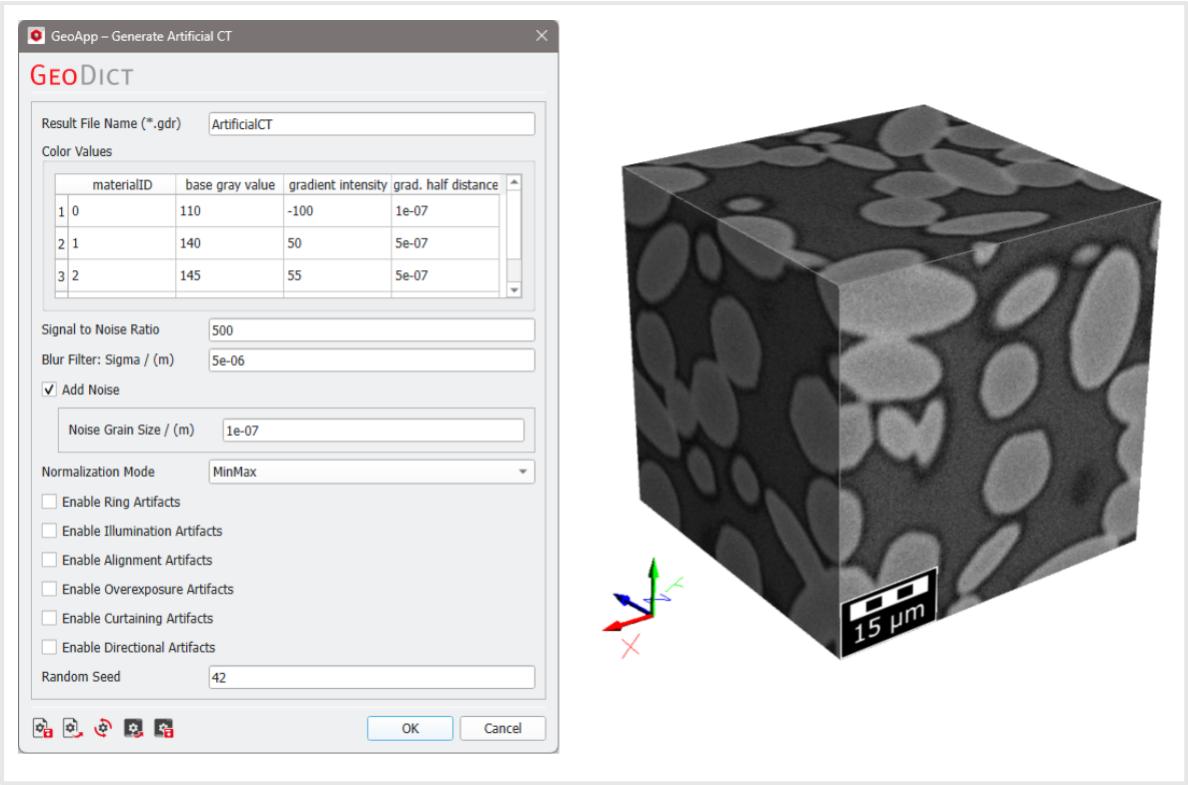
For an example we created a grain structure with GrainGeo Create consisting of spherical objects (material ID01) and ellipsoids (ID 02). Afterwards, the surface was roughened for a more realistic appearance.



Generate an artificial CT-Scan

After generating a structure, an artificial grayscale image can be created including many kinds of artifacts as described in the GeoApp User Guide.

In the example, the different grain material IDs got overlapping gray value ranges and only noise was added.

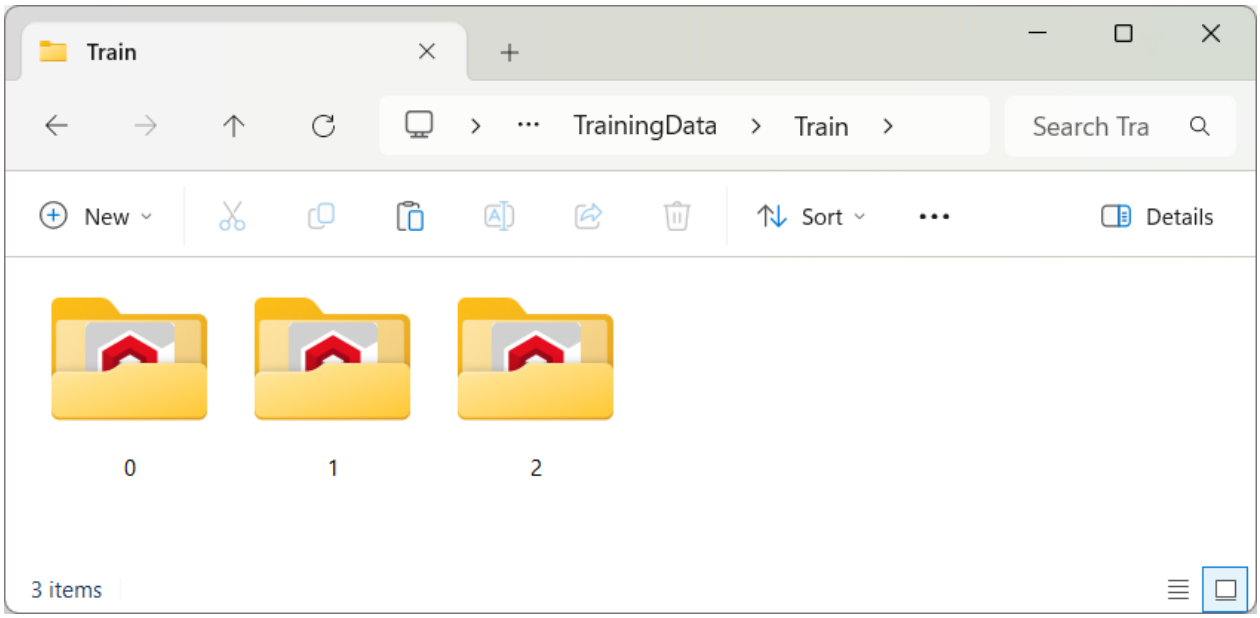


1.2.4 Set-up Training Data Manually

After creating training data without using GeoDict-AI, the available input and output files have to be saved in a defined folder structure. For structure files (*.gdt) generated with the GeoDict structure generator using **Create Training Data,** the required folder structure is produced automatically.

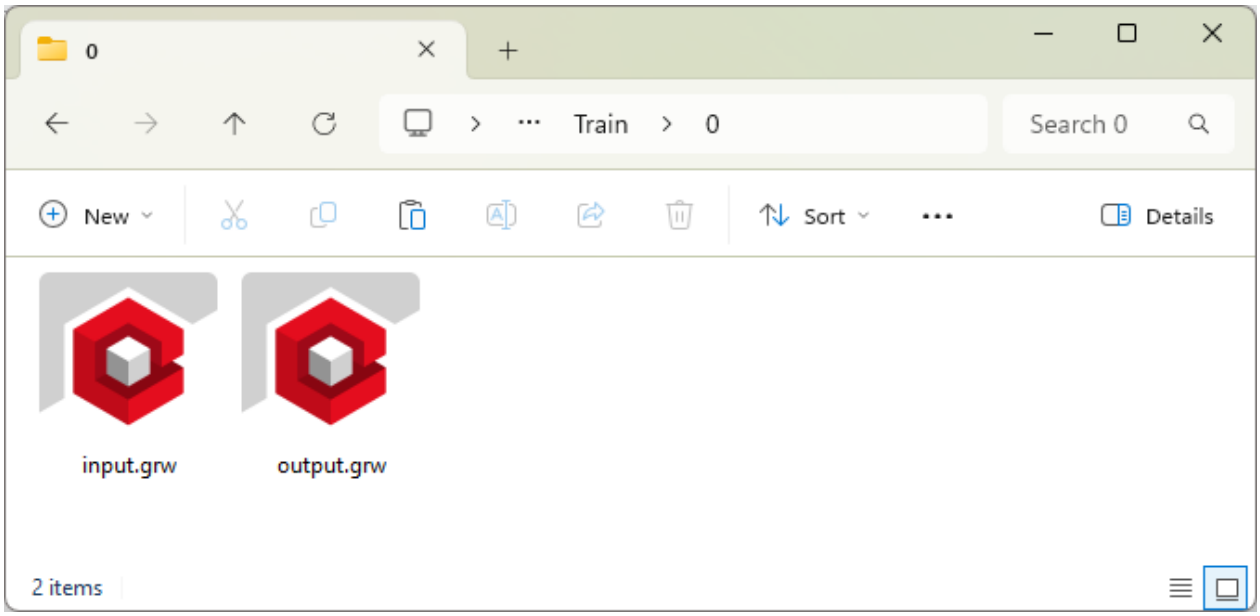
Create an empty folder for the training data. The name of this folder can be chosen freely according to the corresponding project. In the User Guide example, it is called “TrainingData”.

Inside this folder create a folder named **Train**. For each training pair (input and output file) create a folder, which can again be named freely, for example with the scan’s name. In the example we called the folders as it is done when creating training data automatically with numbered folders starting with “0”.

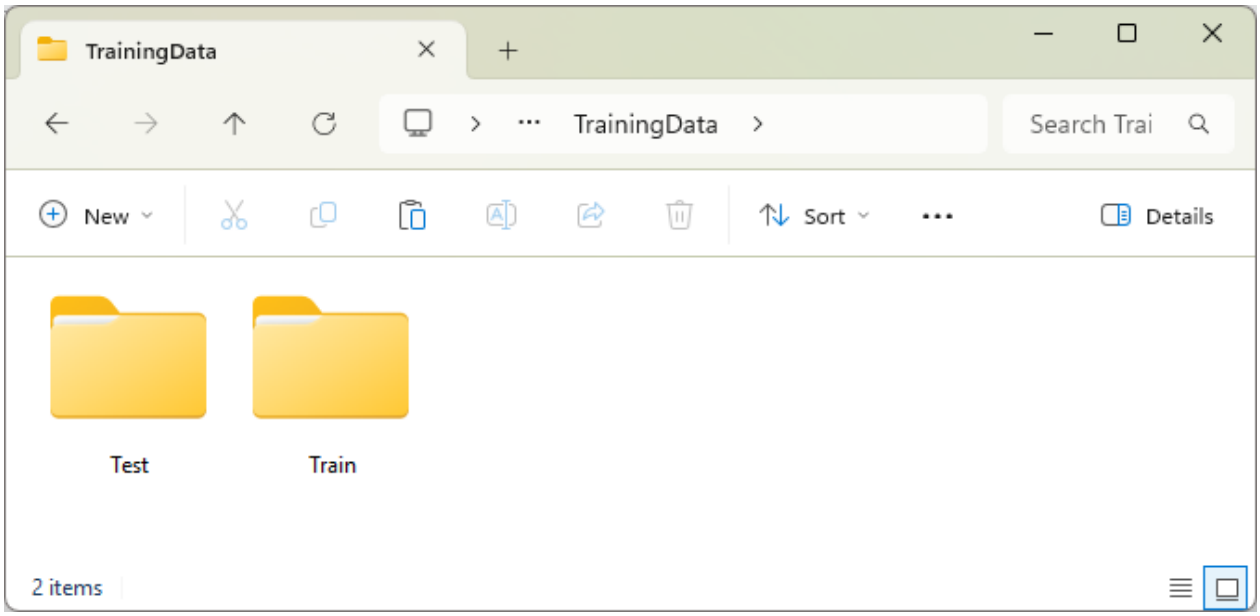


The neural network will learn how to transform the input into the given output. Thus, place the training data pairs in the folders. The training data pairs can have the following file formats depending on the application:

- input.grw and output.grw for enhancing grayscale images,
- input.gdt and output.gdt for identifying different materials or individual fibers in a structure, or
- input.grw & output.gdt for segmenting grayscale images.

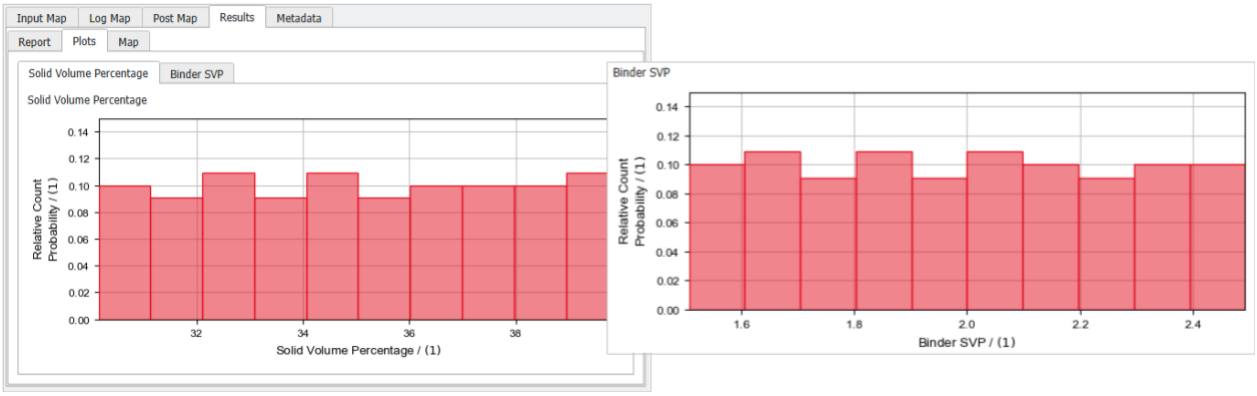


To identify fibers and distinguish between material phases, additionally a folder labeled **Test** can be placed inside the training data folder containing more folders with input.gdt and output.gdt pairs, if enough pairs are available. This data can be used in Validate Performance 84 to verify if the network performs well on structures not included in the training.


TrainingData
TrainingData
Search Trai
New
Sort
Details
Test
Train
2 items

1.3 Design of Experiments

The training of a neural network begins with setting the experiment parameters.



To identify individual fibers or to distinguish different material phases, you can use a so-called generation script and define the parameters for the training data using **Design of Experiments**.

A generation script can be created as shown here 15.

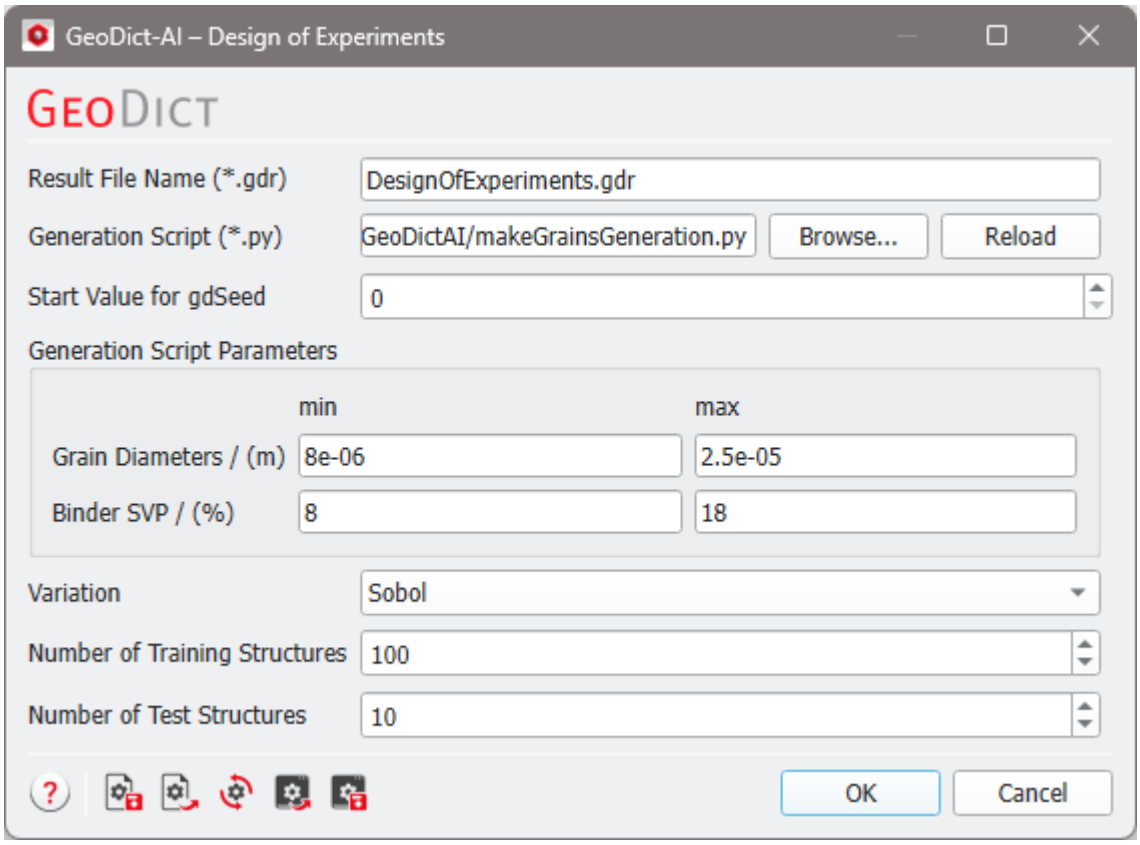
The experiment design results then can be used in **Create Training Data** 33 to generate the input-output-pairs for the training.

The Design of Experiments command:

- Options 27
  Define the parameters for the experiment design.
- Results 31
  View the experiment design results.

1.3.1 Options

To set the experiment parameters, select **Design of Experiments** from the pull-down menu in the GeoDict-AI section. The **GeoDict-AI - Design of Experiments** dialog opens when clicking the **Options**' **Edit...** button



If all parameters are set as desired, click **OK** to close the dialog and in the GeoDict-AI section click **Run**. The generated results are explained here 31.

Result File Name

At the top of the **GeoDict-AI Design of Experiments** dialog, the name for the files containing the design results can be entered for **Result File Name (*.gdr)**. The default name can be kept, or a new name can be chosen to fit the current project.

Generation Script and Generation Script Parameters

For **Generation Script**, browse to the *.py file containing the information to generate the training data. How to create this GeoPy file is explained here 15.

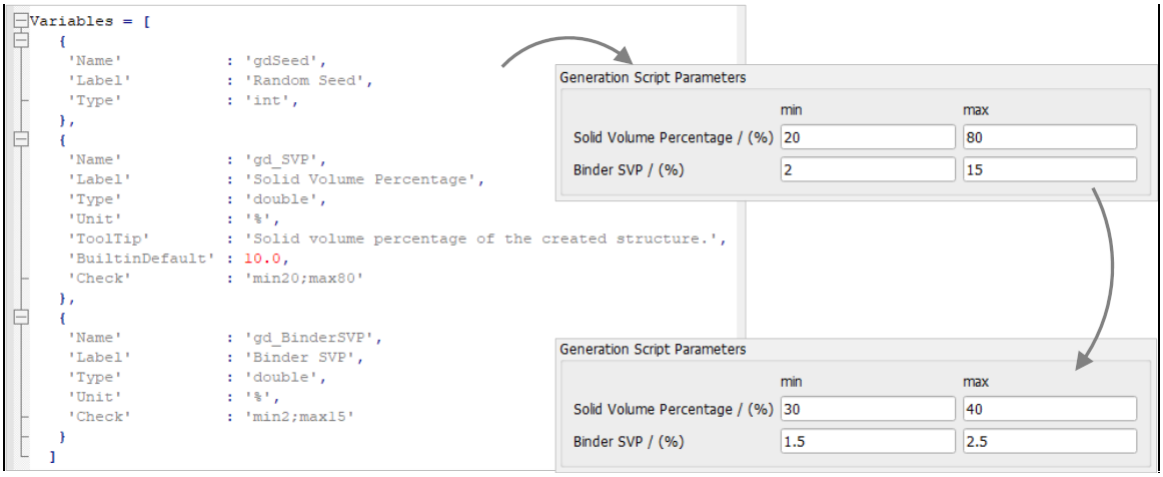
By default the built-in example **makeGrainsGeneration.py** is loaded. To learn more about this example refer to the Create Training Data 35 topic.

The generation script must contain the parameter **gdSeed**. It should be used, to provide a random seed to any stochastic structure generation commands (e.g., FiberGeo or GrainGeo). Enter a **Start Value for gdSeed**. Starting with the entered value, with each structure generation gdSeed is counted one up.

After selecting a generation script, the **Generation Script Parameters** panel appears below the gdSeed value. If the generation script contains variables other than the

gdSeed parameter, here, the allowed value ranges for these parameters can be given in the **min** and **max** parameter boxes.

In the example below, the selected **Generation Script** contains the parameters gdSeed, gd_SVP and gd_BinderSVP. According to the entered check values, which are taken automatically in the Design of Experiments dialog, the solid volume percentage should be chosen between 20% and 80% and the Binder SVP should be chosen between 2% and 15%. In the example, we then decided, that the solid volume percentage will be varied between 30% and 40%, and the binder solid volume percentage will vary between 1.5% and 2.5%.



Variation

Select a **Variation** method for the script parameters to determine how the parameters are chosen from the entered value ranges. The available number generation methods are **Sobol** and **Random**.

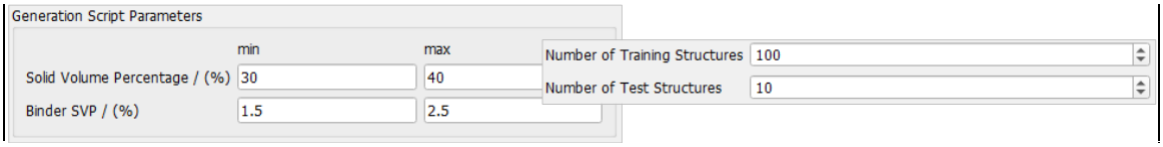
Random will draw independent random samples in the given parameter ranges. The start value for gdSeed will be used to initialize the random number generation. While all other settings are left the same, the same **gdSeed** start value always generates the same parameter sequences. Accordingly, different seeds generate different parameter sequences.

Sobol uses a so-called sobol sequence for sampling which generally results in more uniform covering of the n-dimensional parameter space. **Sobol** does not depend on gdSeed. To learn more about this algorithm refer to Sobol, 1967.

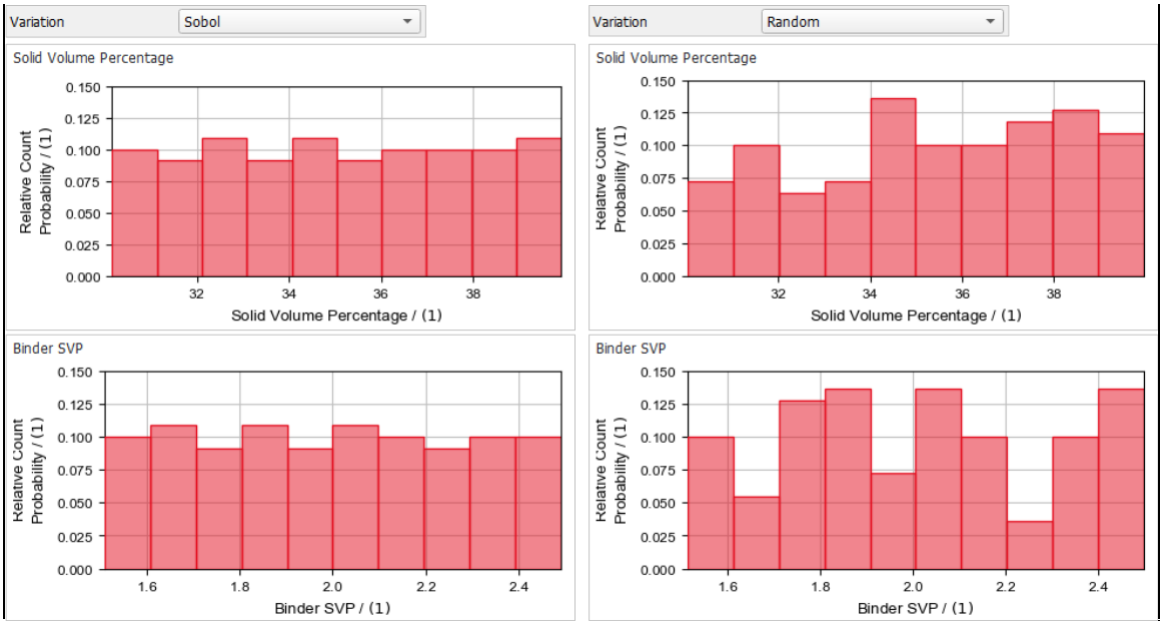
In the following example, observe the differences between the two **Variation** options **Sobol** and **Random**.

The example script had three parameters. The **Start Value for gdSeed** was set to 5 and the parameter ranges were set to 30-40 for **Solid Volume Percentage** and 1.5 – 2.5 for **Binder SVP**.

The **Number of Training Structures** was set to 100 and the **Number of Test Structures** were set to 10.



This results in the following plots for the two different **Variation** options. Observe how the parameter values are selected uniformly in the given ranges, if **Sobol** is selected, while they are chosen randomly, if **Random** is selected.



Number of Training and Test Structures

The **Training Structures** will be used later to train a neural network. For each parameter combination an input and an output file are generated. The neural network will learn how to transform the input file in the given output file. Learn more about input and output files here 35. Select the **Number of Training Structures**. The more structures used, the more the neural network can learn in the training. The minimum needed number depends on the number of varying structure parameters, but in most cases 100 training structures lead to good results.

The **Test Structures** can be used to validate a neural network, that was trained on the training structures. They are created with the same generation script but with different parameter combinations within the given boundaries. If the neural network performs well on these test structures, it should also deliver good results on segmented scans with the corresponding statistical properties. The performance is validated, by applying the trained network to the input files and comparing the results to the corresponding output files. Learn more about input and output files here 35. Select the **Number of Test Structures**. The more structures used, the better for the neural network validation. In most cases with 10 test structures the network can be validated 84 well.

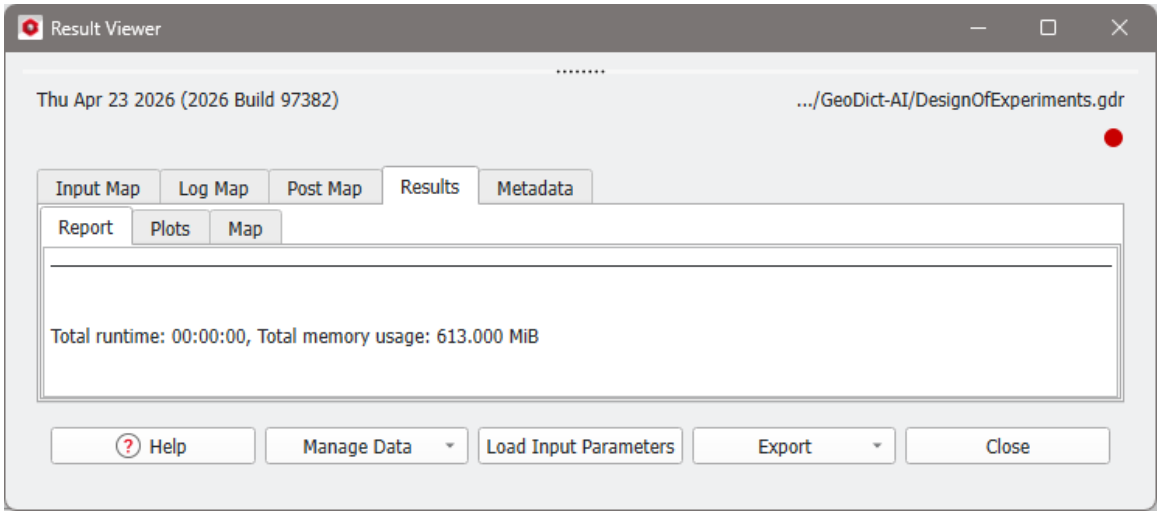
However, for testing of the generation script it is recommended to choose a small **Number of Training** and **Test Structures**, e.g., 1-5.

1.3.2 Results

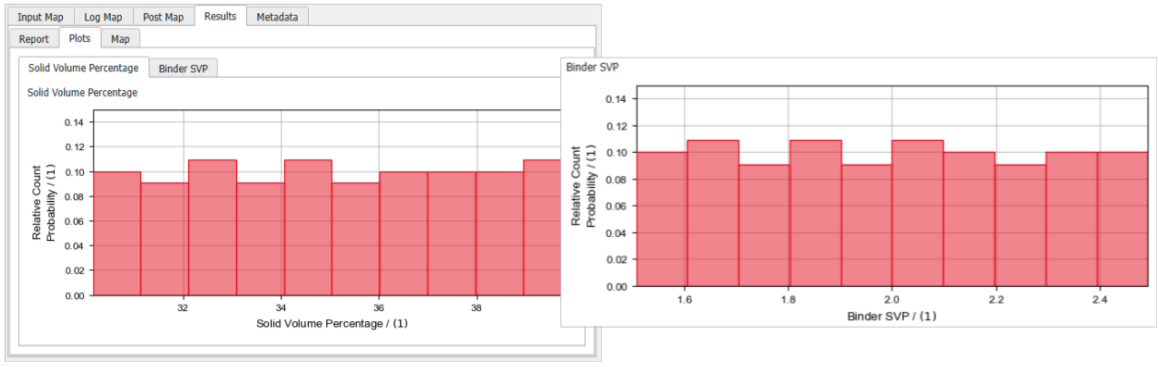
Running **Design of Experiments** produces a GeoDict result file (*.gdr) and a result folder with the same name. Both are saved in the chosen project folder (**File** → **Choose Project Folder...** in the menu bar).

Result Viewer

The GeoDict **Result Viewer** opens for the result file and only the performance of the experiment design is shown in the **Results – Report** subtab.



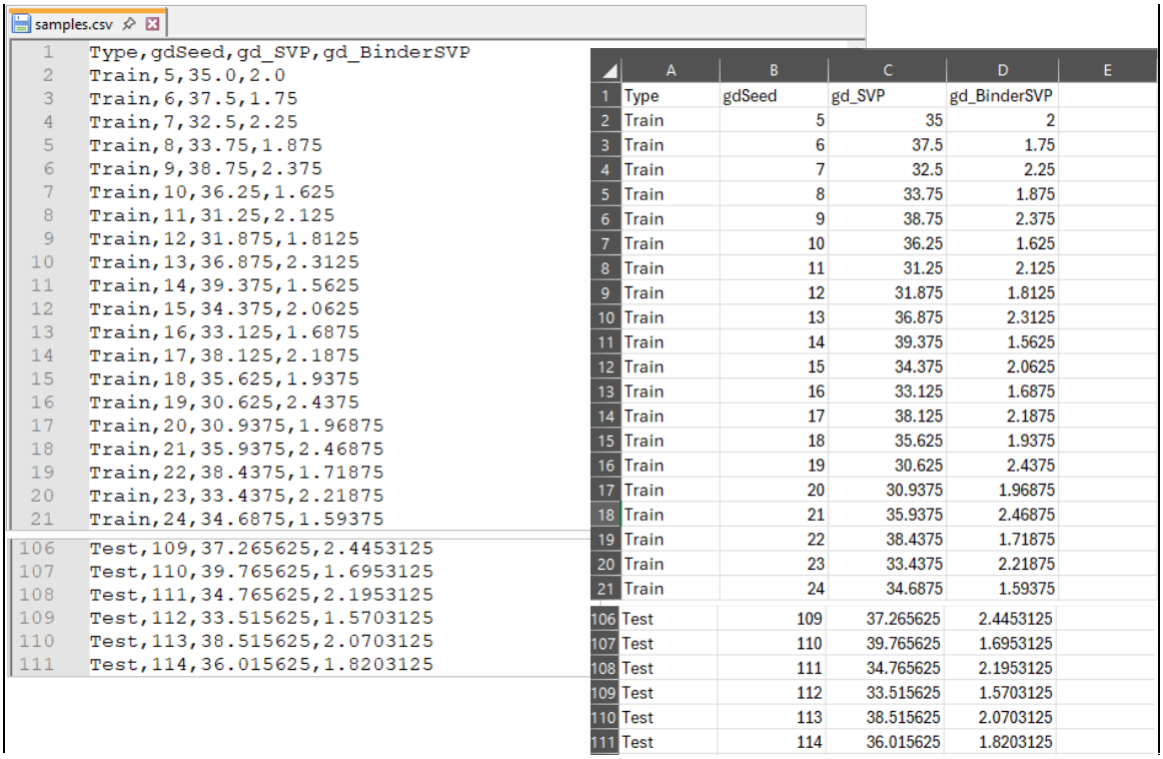
If the **Generation Script** contains other parameters besides **gdSeed**, the **Results – Plot** subtab provides plots for all these parameters. The plots show the **Relative Count Probability** for each value in the given ranges[28].



Result Folder

The result folder contains the **samples.csv** file. This is the file later used to Create Training Data 34, and allows to check, if the parameter combinations produced from the given parameter ranges are reasonable for the desired training data, before starting the time-consuming generation process.

The **samples.csv** file can also be opened in Excel or a text editor to review the content.



The first column in the **samples.csv** always contains the **Type** for the generated value combinations. The possible types are **Train** and **Test**. The number of rows then depends on the **Number of Training Structures** (**Train**) and the **Number of Test Structures** (**Test**) entered in the **Design of Experiments Options** dialog.

In the second column the values for **gdSeed** can be found starting with the **Start Value for gdSeed**.

If there are also other parameters in the **Generation Script**, a column with random values in the given ranges is produced for each of them.

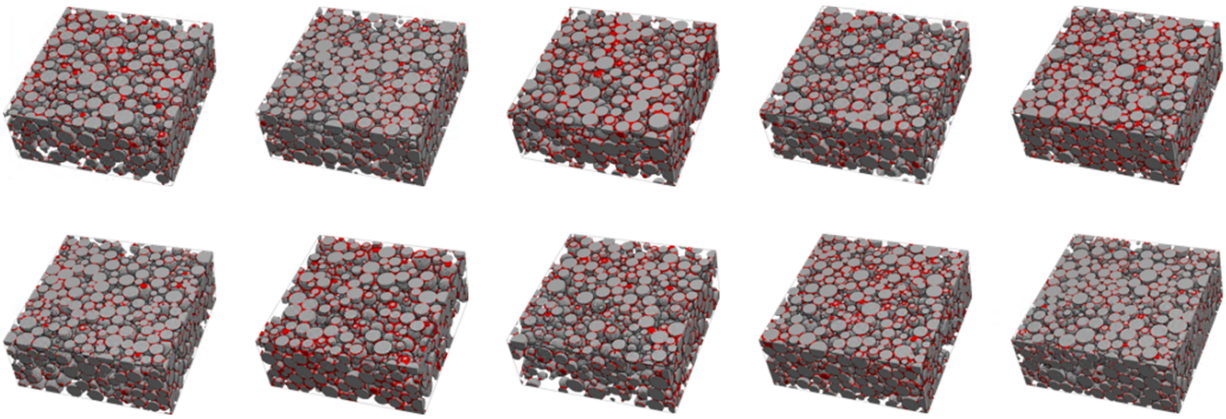
1.4 Create Training Data

For the training of a neural network you need training data consisting of input-output pairs.

To identify individual fibers or to distinguish different material phases, you can use a so-called generation script with **Create Training Data** for the data generation. A generation script can be created as shown here 15.

Define the parameter ranges and number of input-output pairs with **Design of Experiments** 27.

Below find an example for 10 possible output files for identifying binder in a grain structure:

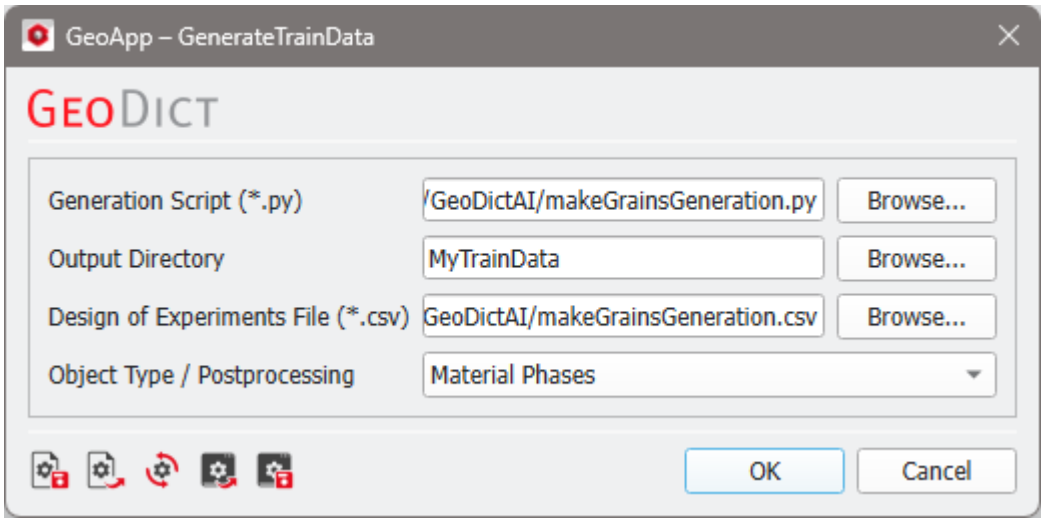


The Create Training Data command:

- Options 33
  Define the parameters for the training data creation.
- Results 35
  Have a look at the created training data.

1.4.1 Options

In the GeoDict-AI section select **Create Training Data** from the pull-down menu. The **Generate Train Data Parameters** dialog opens when clicking the **Options**' **Edit...** button in the GeoDict-AI section.



If all parameters are set as desired, click **OK** to close the dialog and in the GeoDict-AI section click **Run**.

The results are explained here 35.

Generation Script

For **Generation Script**, browse to the *.py file containing the information to generate the training data. How to create this GeoPy file is explained here 15. Use the same file as for the corresponding Design of Experiments 28.

Output Directory

For the **Output Directory**, browse to an empty folder to store the resulting training data or create a new folder. The folder must be empty, because it is used to train the neural network 41.

Design of Experiment File

Browse for the created **Design of Experiments File (*.csv)** to set up the experiment parameter ranges and number of training and test structures. This file is found in the result folder of the Design of Experiments 31 run.

Object Type / Postprocessing

Two different **Object Types** can be selected to define the **Postprocessing**.

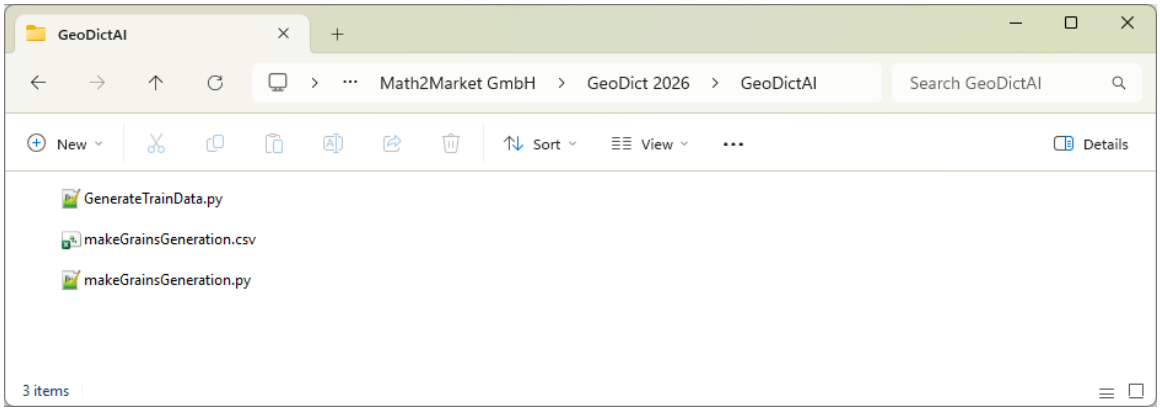
Select **Material Phases,** if material phases should be distinguished, e.g., grains and binder or circular fibers and trilobal fibers. For this, the **Generation Script** must produce the files **input.gdt** and **output.gdt** as described here 18.

Choose **Fibers (centerlines),** if the individual fibers should be identified. With this option, **Create Training Data** will perform a postprocessing, identifying the fiber's centerlines and producing the files input.gdt and output.gdt, automatically.

Know how! Note that GeoDict can identify individual grains as well. For this purpose, use the GrainFind module, where grains are identified using the watershed algorithm.

Default Example

By default, for **Generation Script** and **Design of Experiments File**, two example files are loaded. Find the files **makeGrainsForTraining.py** and **makeGrainsForTraning.csv** files in the GeoDict installation folder. Browse for the GeoDict **installation folder** and find the files inside the **GeoDictAI** folder.

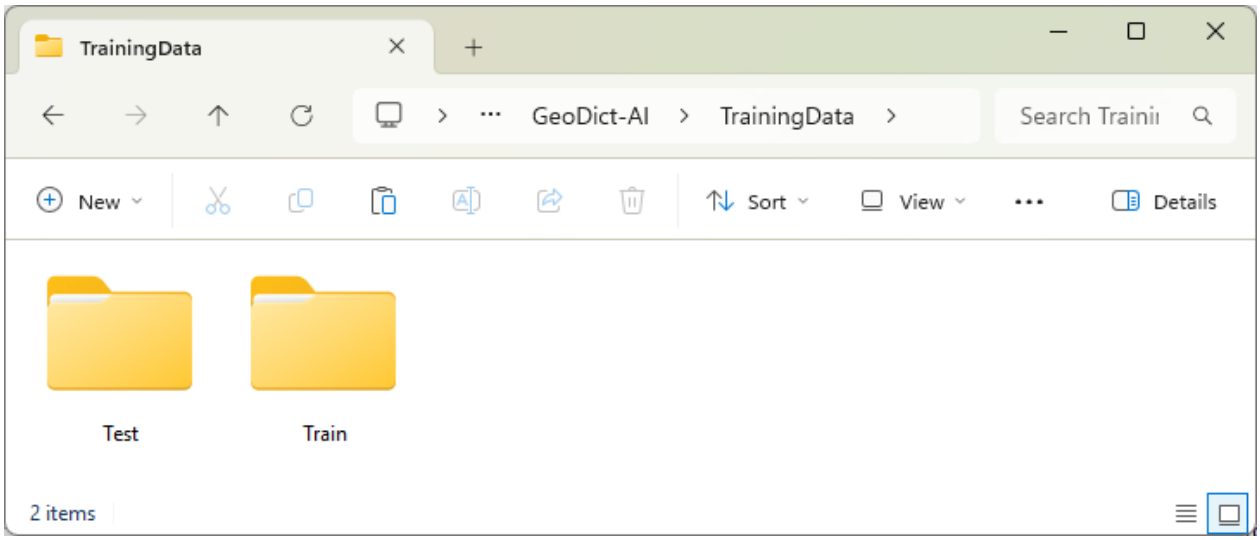


The generation script example generates a grain structure with overlap, adds binder and saves **input.gdt** and **output.gdt.** Open **makeGrainsForTraining.py** in a text editor to examine the structure of the example **Generation Script**.

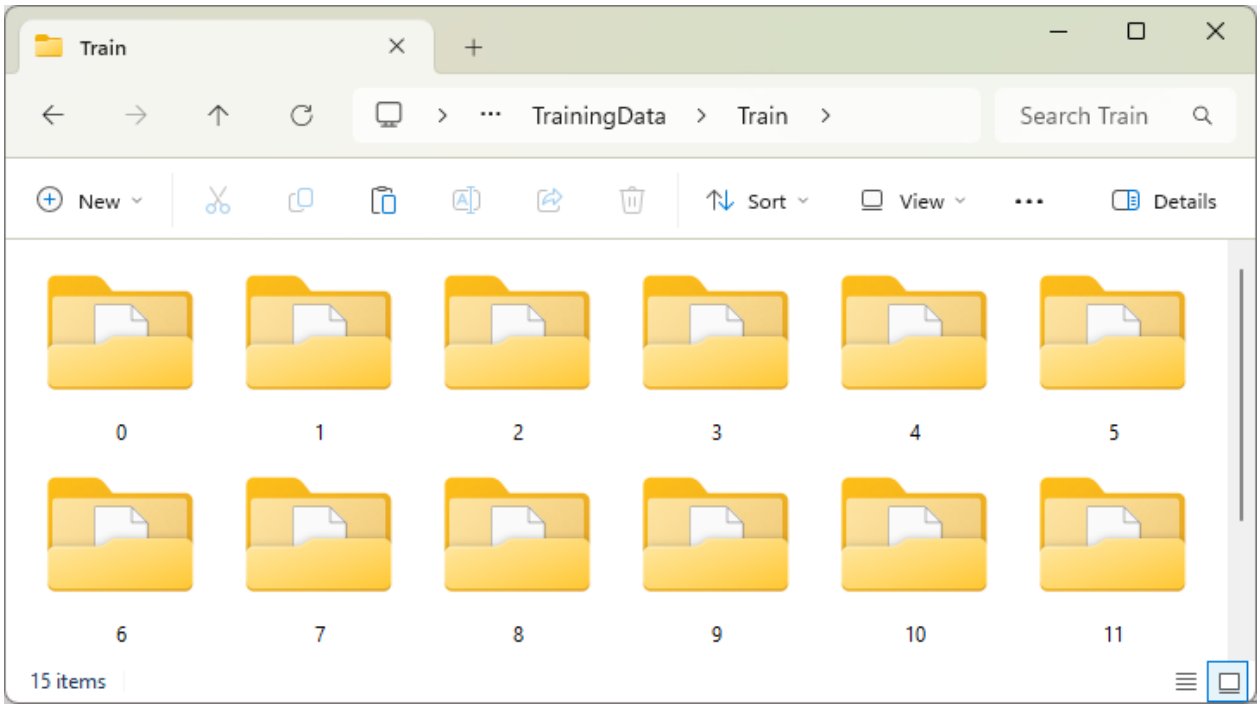
The training settings, as the variable values and number of training structures are controlled by the **makeGrainsForTraining.csv** file. It can be opened in Excel or another text editor.

1.4.2 Results

First the **Training Structures** are generated, followed by the **Test Structures**. They are placed in the subfolders **Train** and **Test** inside the selected **Output Directory**.



For the training only the folder **Train** will be used. The folder **Test** can be used to validate the performance of the network.



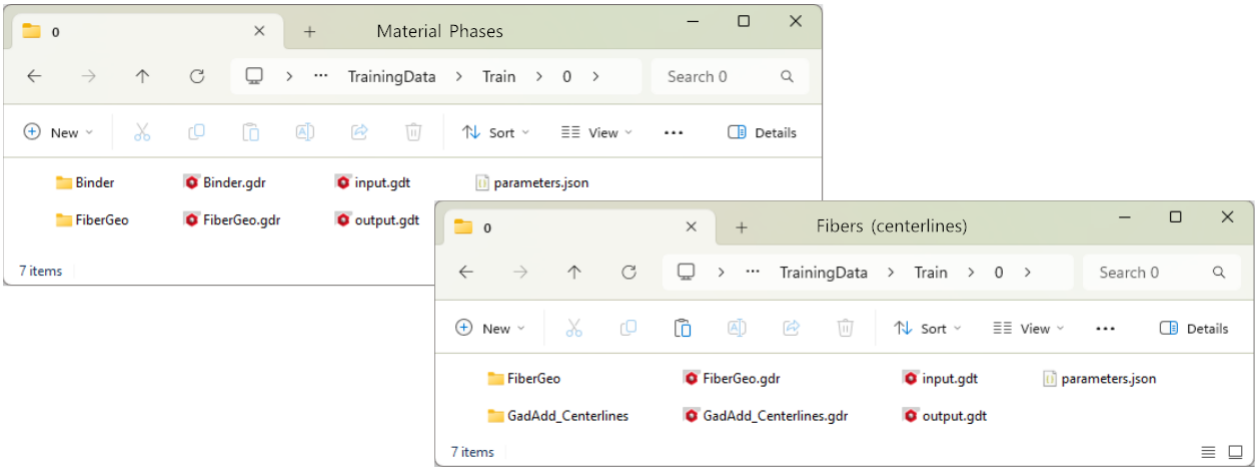
Each of the subfolders (e.g., 'Train/0/') contains the results produced by the **Generation Script** and a pair of GDT files called **input.gdt** and **output.gdt**.

If **Material Phases** is selected for **Object Type / Postprocessing**, the input and output files must be generated by the **Generation Script**. In the **input.gdt** then, the **Target Material** cannot be distinguished, while in the **output.gdt**, the **Target Materials** have their own material IDs.

The two files are generated automatically if **Fibers (Centerlines)** is selected instead. Then, the **input.gdt** contains only the fibers, which are then assigned to material ID 01, while in the **output.gdt** the centerlines in the fibers are assigned to material ID 02.

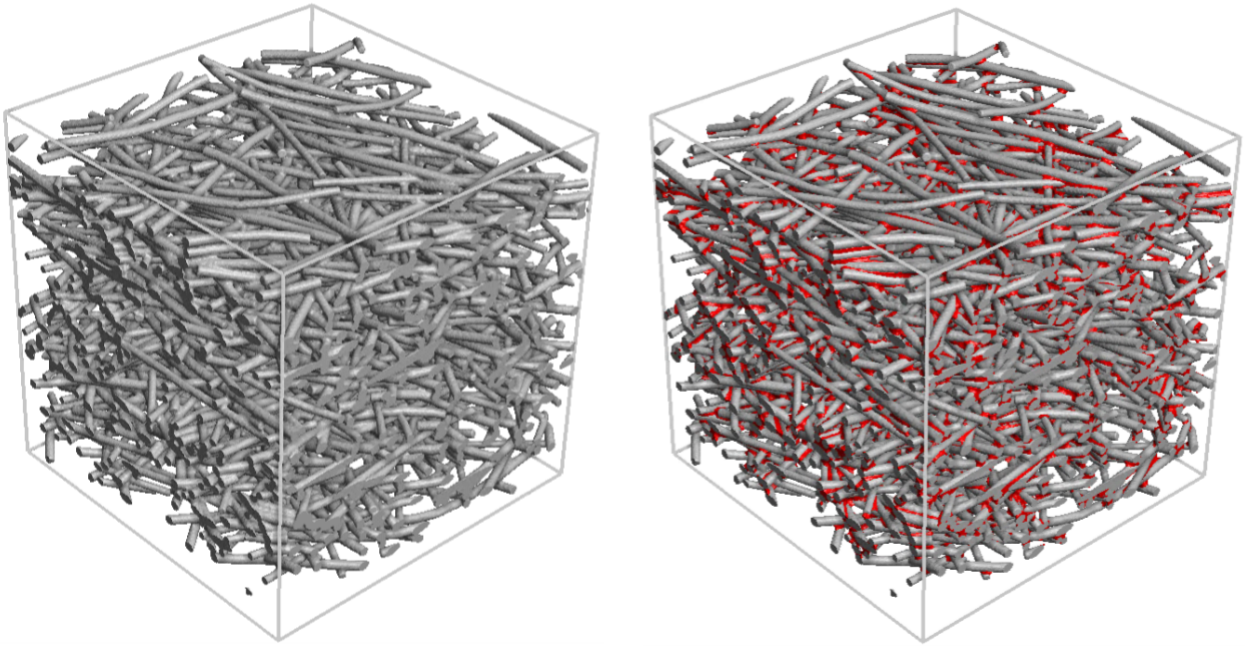
Later in training, in both cases, the neural network learns how to produce an output from an input file by assigning those voxels with the corresponding material ID.

The macro parameters for the training structure can be viewed by loading the **parameters.json** file into a text editor.

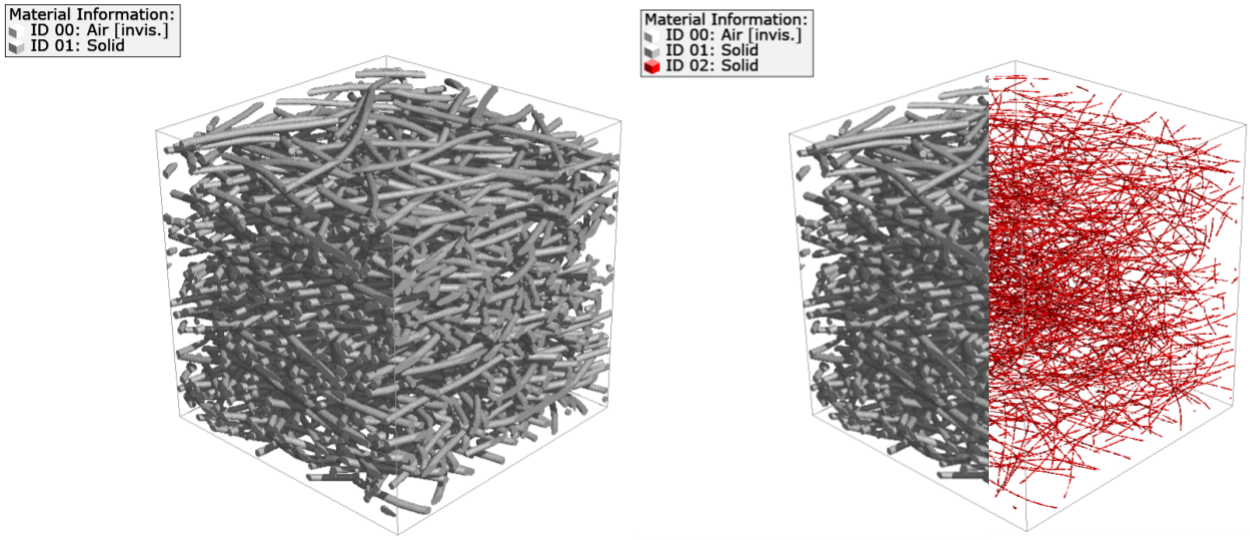


To verify that the input and output files are as intended, load them to GeoDict by selecting **File → Open Structure (*.gdt, *.gad) ...** from the menu bar.

Below find an example for differentiating between two material phases:



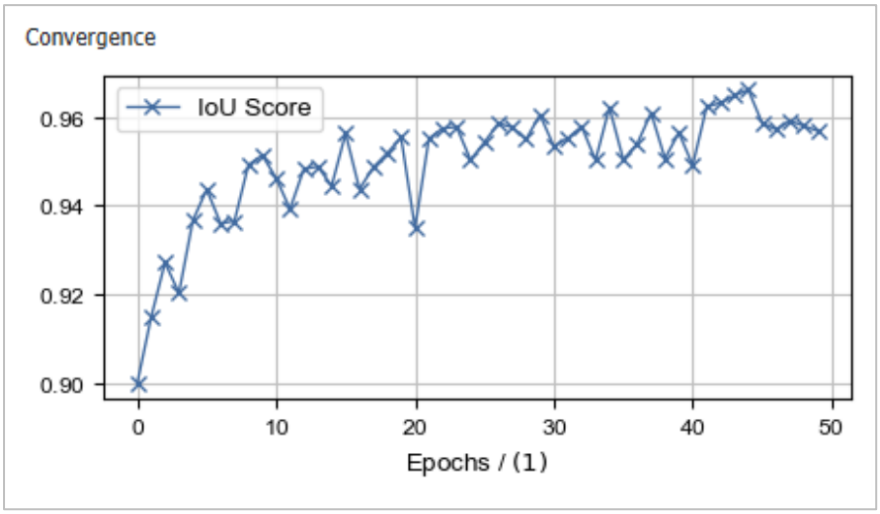
Below find an example for identifying individual fibers. In the output file on the right Material ID 01 is not visualized in the right part of the structure to highlight the center lines (ID 02).



Alternatively, if input and output files are not generated with a generation script, create the folder structure manually and place the input.gdt and output.gdt in subfolders folders inside the Train and Test 24 folders.

1.5 Train Neural Network

After creating training data either using **Create Training Data** 33 or providing it manually 24, you can use this data to train your own neural network. Learn more about the underlying theory in the chapter Neural Network Training 8.



The Train Neural Network command:

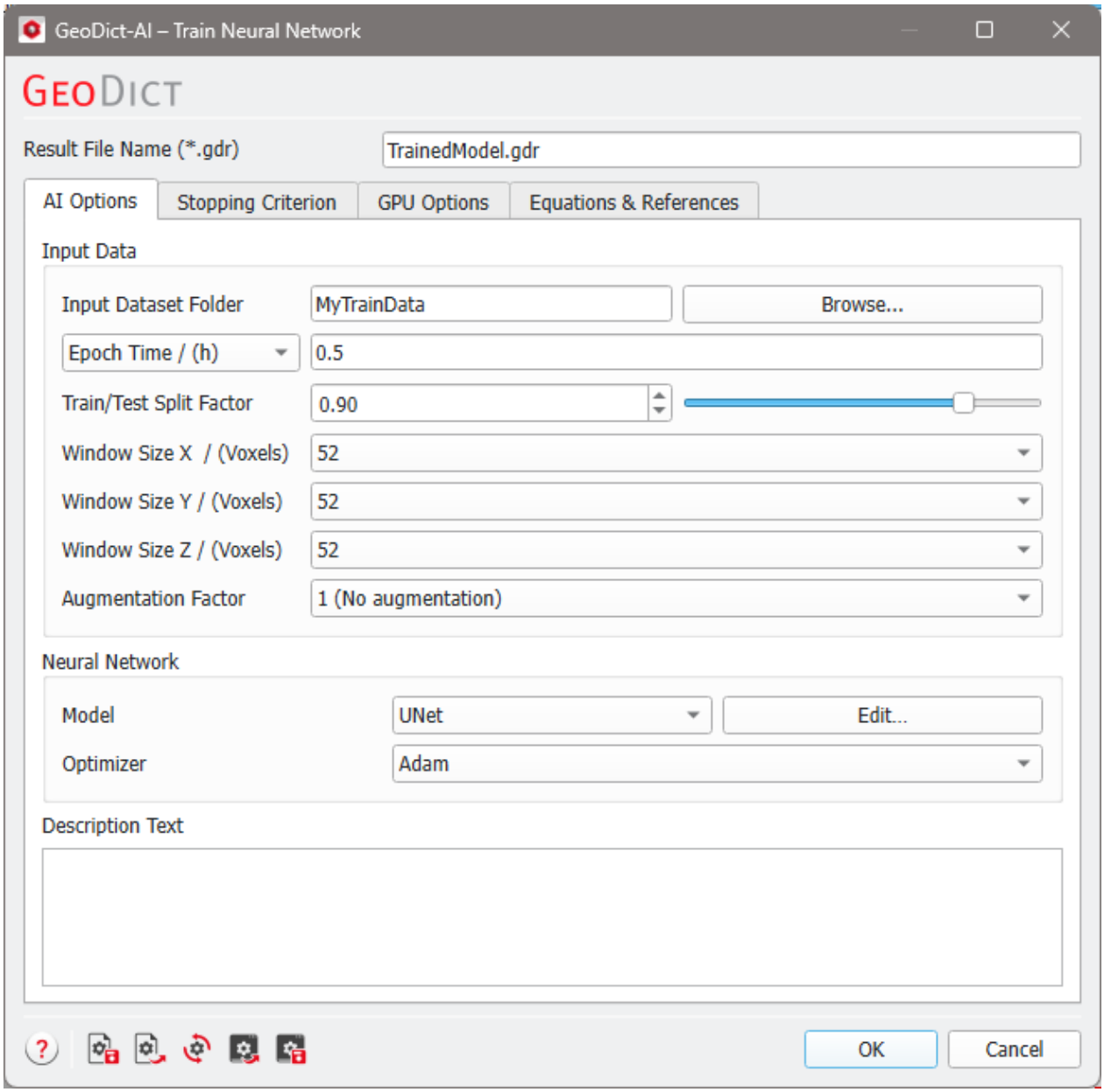
- Options 39
  Define the parameters for the training.
- Run Training 50
  Learn how to verify that a training runs well.
- Results 54
  Learn about the training results.

1.5.1 Options

In the GeoDict-AI section, select **Train Neural Network** from the pull-down menu. The **GeoDict-AI - Train Neural Network** options dialog opens after clicking the **Options' Edit...** button in the GeoDict-AI section. When opening this dialog the first time, it can take some time as GeoDict checks for available GPUs.

At the top of the dialog, the name for the files containing the training results can be entered in the **Result File Name (*.gdr)** box. The default name can be kept, or a new name can be chosen to fit the current project.

As usual in GeoDict, the training process produces a .gdr file and a corresponding result folder of the same name. The result file (*.gdr) contains information about the training process. The corresponding result folder contains the trained model.



Topics of this section

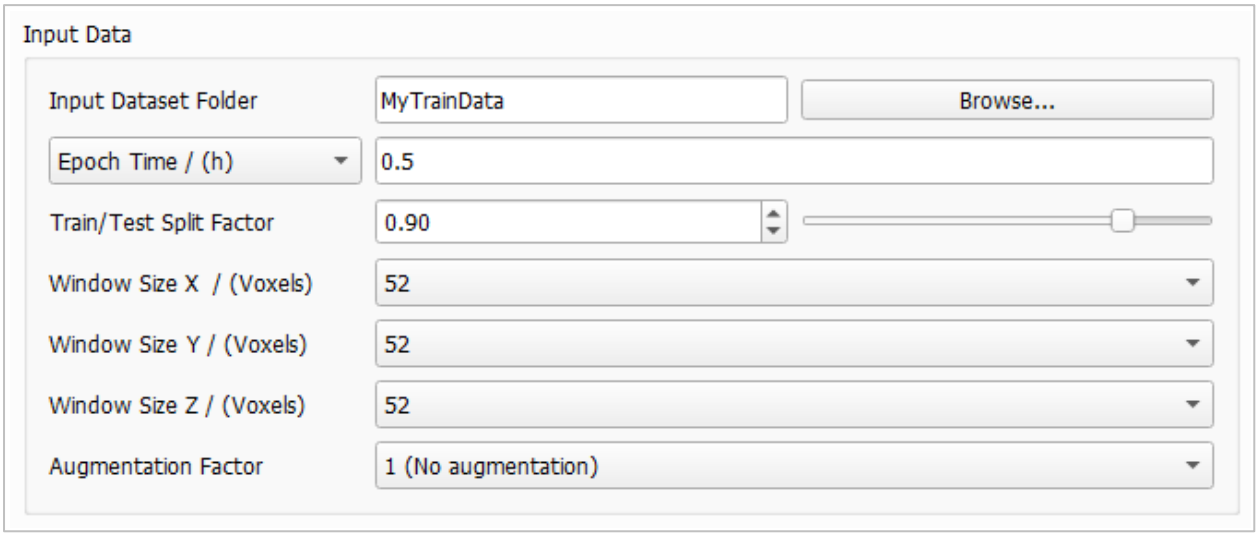
- AI Options 40
  Define the parameters for the training, as input data, window size and UNet parameters.
- Stopping Criterion 48
  Define when the training should stop.
- GPU Options 49
  Select a GPU to use and enter a batch size.

AI Options

Under the AI options set-up the training of the neural network.

Input Data

In the **Input Data** panel specify the training data.



Input Dataset Folder

Browse for the **Input Dataset Folder** containing the subfolders **Train** and **Test**. This folder is filled with training data when running Create Training Data 34 or manually 24.

Epoch Time and Downsampling Factor

Define the duration of one epoch, either directly by selecting **Epoch Time** or by selecting **Downsampling Factor**. This defines how often an intermediate neural network is saved to the result folder.

Also, considering every possible window position in the structure as described for Window Size 43 leads to a large amount of data. The amount of data is reduced by the given epoch time or the downsampling factor.

For **Epoch Time** enter a time, usually between 0.1 and 0.5 hours. Then, the entered value will be the exact time of the first epoch 51 to find out the needed number of samples (or window positions). All following epochs will use the same number of samples, such that all epochs will need approximately the same time. The maximum number of epochs can be entered under the Stopping Criterion 48 tab.

The **Downsampling Factor** is a value in [0,1] which is used to directly reduce the data amount during training.

After loading all the data defined by the window size, GeoDict-AI will ignore some of the data samples depending on the factor, e.g., if 0.1 is specified, a random 10% subset of the data will be retained. The data is sampled uniformly from all structures.

For the training in most cases a downsampling factor between 0.01 or 0.001 is recommended. However, this depends on the amount of training data. The larger the amount of training data, the smaller the **Downsampling Factor** should be chosen for a higher frequency of intermediate results.

The larger the downsampling factor, the longer the runtime of one epoch. But for a larger factor, less epochs are needed to reach the same result. This means, while the downsampling factor affects the training time of one epoch, it does not significantly affect the overall training time.

Know how! One epoch should always run at least some minutes, since saving and validating of the model always needs the same time and is done in each epoch. Increase the downsampling factor if the epoch duration is only a few seconds.

Reaching an epoch runtime of about 30 minutes is a good reference point. For a complex training also epoch runtimes of a few hours can be fine.

Note! The parameters changed in comparison to GeoDict 2023 and older. The parameter “Window Stride” already reduced the number of window positions. Thus, a higher downsampling factor was recommended. When loading the input parameters of an older GeoDict result file (*.gdr) the downsampling factor is decreased, automatically.

Train/Test Split Factor

Before starting with the training, the training structures (images) - generated with Create Training Data 33 or manually 24 - are split into all the windows defined by Window Size 43. After permuting these windows randomly, the **Train/Test Split Factor** determines the number of windows which are used for the training and how many are used for the evaluation during training. This way, the structures (images) from the **Train** folder are divided in train and test data and all training structures are represented in the train data as well as in the test data.

The default value 0.9 means that 90% of the windows are used for training and 10% for evaluation. The 10% of the training data are used for internal tests during the training, i.e., to compute the validation loss 52 by applying the neural network to these structures (images) at the end of each epoch. This is necessary to have a measure on how well the network can generalize what it has learned to the new data. In most cases, the default of 0.9 for the split factor produces good results.

Note! The structures (images) from the Test 35 folder are not used for the training at all. They are meant to be used to validate the performance 85 of

the neural network after training.

Window Size

The **Window Size** determines the size (in X, Y and Z) of the sampling window in voxels. The training data consists of several input and output files[24]. For the training cutouts from corresponding **input** and **output** files are taken in the same location with the given window size. Thus, all network's decisions are based on these cropped data pairs alone. The neural network will try and learn to transform what it sees in the input window to what it observes in the output window. The **Window Size** determines how much context the network obtains for its decisions. Larger values improve the performance, while increasing training time and GPU (memory usage).

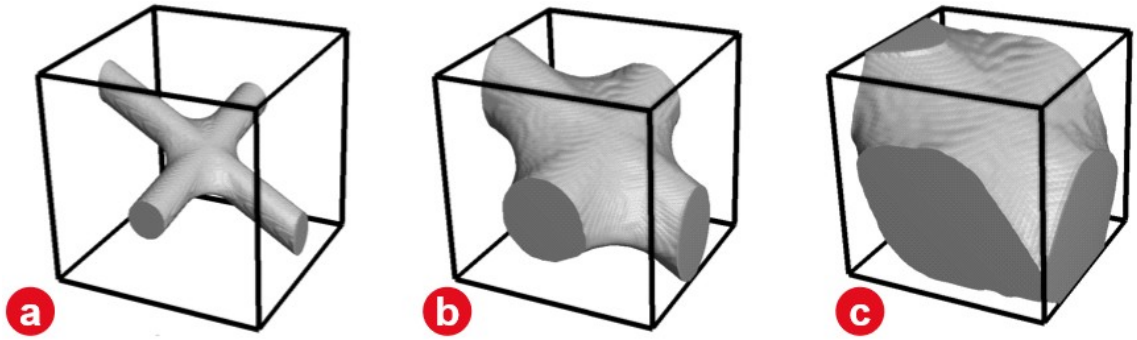
Any possible placement of the window within the structure corresponds to a potential training sample. However, considering all possible locations during one epoch is usually redundant and prohibitively expensive in terms of runtime. To reduce this large amount of data, the parameters Downsampling Factor[41] or Epoch Time[41] are used.

The **Window Size** must be large enough to capture an entire important feature, e.g., a fiber crossing with binder. This way it is possible to distinguish the different materials only from the window.

For a fiber structure, for example, it is recommended to choose the window bigger than twice as large as the diameter of the largest fiber. This allows two touching fibers to be fully contained in a single window. This way the network can correctly identify the geometric configuration, as two fibers crossing or touching and hence, make an informed decision about the binder placement around the touching point.

Consider differentiating binder and fibers only based on a cutout in the selected window size. If a human cannot distinguish the material phase from the geometry seen in the cutout window, it is unlikely that a neural network is capable to. In such cases, the window size must be increased so the neural network can "see" the two fibers approaching each other and trace them through the intersection even with the large amount of binder obscuring the actual intersection point. If it is humanly possible to identify the binder, the network can also learn that. Otherwise, the window size must be increased.

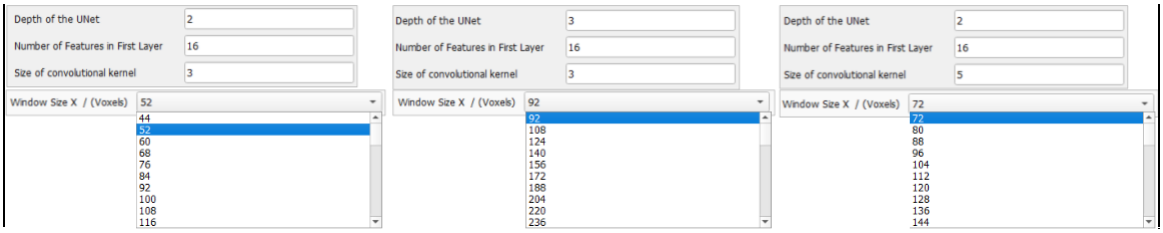
In the next figure, different window sizes are considered. Observe how in **a)** it is easy to distinguish binder from fibers, as the window size is larger than twice the fiber diameter. In image **b)** it is already hard, but still possible with a window size only a little more than twice the fiber diameter. However, in a cutout even smaller than twice the fiber diameter as in **c)** it will be nearly impossible to identify the fibers.



The aspect ratio of the window can be adjusted to the corresponding structure, e.g., for a flat textile structure or for materials which are anisotropic. For example, in the gas-diffusion-layer of a fuel-cell the fibers are usually oriented mainly in the X/Y-plane. Then, a smaller window size in Z-direction is reasonable. For isotropic materials, however, it is recommended to set the window size for all three directions to the same value.

The possible values for **Window Size** also depend on the **Depth** selected for the **UNet** model. In each convolution 1 voxel is lost at each border of the cutout. As there are two convolutions in each layer of the UNet, four voxels are subtracted from the window size in each direction (compare to the UNet diagram 8). The resulting values are divided by two in each max-pool operation. The selected Depth 45 determines the number of applied max-pool operations.

Select the **Window Size** from the pull-down menu. When changing the **Depth** or the **Size of Convolutional Kernel** in the UNet 45 dialog, the pull-down menu for window size is automatically updated and the smallest available window size is selected.



In the following table find possible window sizes for reasonable depth values (and a kernel size of 3). While the pull-down menus also offer larger values, in most cases a window size larger than 200 will not be possible with the given hardware.

Depth	Possible Window Sizes
1	20, 24, 28, 32, 36, 40, 44, 48, 52, 56, 60, 64, 68, 72, 76, 80, 84, 88, 92, 96, 100, 104, 108, 112, 116, 120, 124, 128, 132, 136, 140, 144, 148, 152, 156, 160, 164, 168, 172, 176, 180, 184, 188, 192, 196, 200, 204,...
2	44, 52, 60, 68, 76, 84, 92, 100, 108, 116, 124, 132, 140, 148, 156, 164, 172, 180, 188, 196, 204, ...

Depth	Possible Window Sizes
3	92, 108, 124, 140, 156, 172, 188, 204, ...
4	188, 220, 252, 284, 316, 348, ...

Augmentation Factor

The **Augmentation Factor** allows to effectively increase the amount of training data.

For the default of **1 (No Augmentation)**, each window is seen once during a training epoch.

Setting it to **48 (Full data augmentation, all rotations)**, the window will also be mirrored and rotated, effectively increasing the data set size by factor 48 with the corresponding increase in runtime.

The option of factor **16 (No rotations out of the XY plane, for anisotropic geometries)** which is specifically for structures where fibers are mainly oriented in the XY-plane. Then, only mirroring and rotation in the XY-Plane is performed.

However, values higher than 1 are mostly useful for a small data set, e.g., a single structure, to get as much information as possible from it.

Neural Network

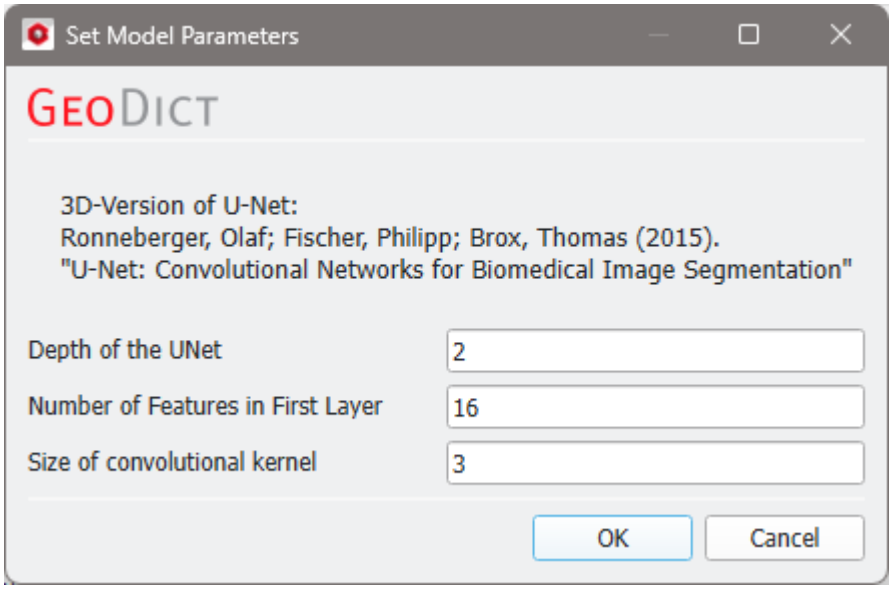
In the **Neural Network** panel define the parameters for the network.

Model

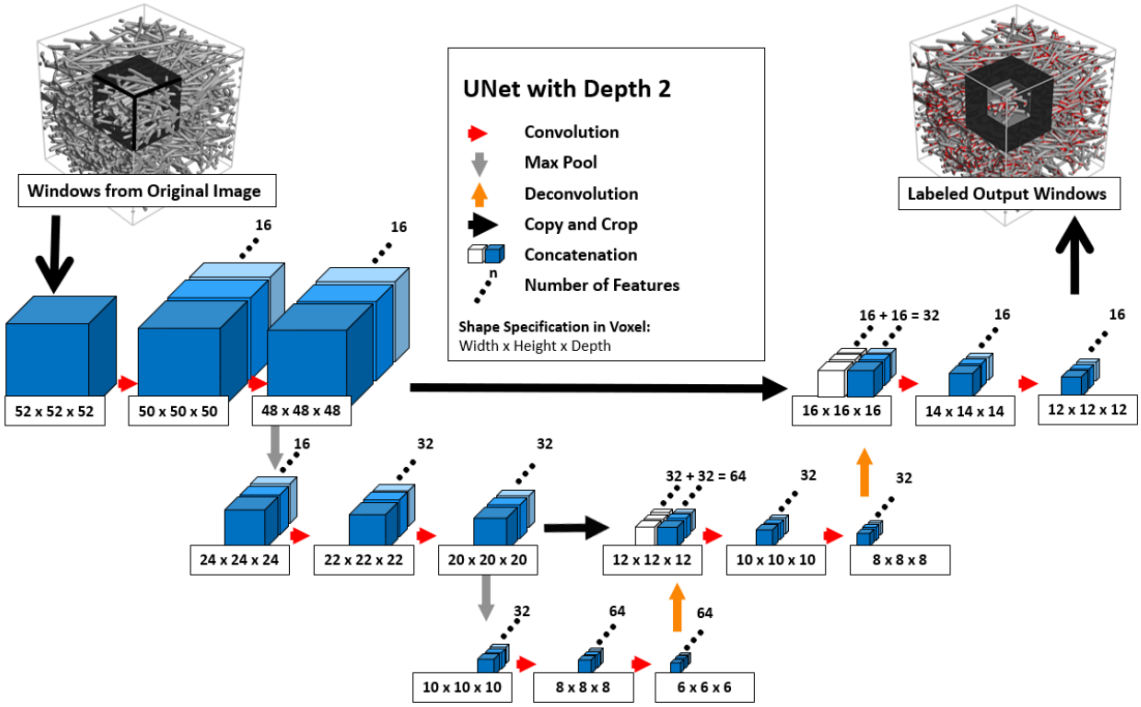
The pull-down menu for **Model** contains the neural network architecture that is being used for training. Currently only the **UNet** is available, which performs very well for segmentation tasks. In future releases, more models will be added. Clicking **Edit** opens the **Set Model Parameters** dialog.

In this dialog the reference Ronneberger, Fischer and Brox, 2015 describes the underlying architecture of the **UNet**. Increasing the parameters **Depth of the UNet** and

Number of Features in First Layer improves the performance of the network, but also significantly increases the training time and the amount of used GPU memory.



The **UNet** has a U-shape (hence the name) with a constricting and an expanding branch on the left and right respectively. The diagram below shows a UNet with depth 2, 16 features in the first layer and a window size of 52 voxels in X-, Y- and Z-direction. Find a detailed description of the diagram here[8].



The **Depth of the UNet** corresponds to the number of layers in each of the branches. Increasing the depth massively increases the complexity of the network, but also the amount of abstraction that it can perform. In Ronneberger, Fischer and Brox, 2015[99]

the UNet operates in 2D, allowing the authors to use a depth of 4, but since GeoDict-AI performs 3D analysis the limits of GPU memory are reached quickly when increasing this parameter. In general, only 2 or 3 are possible and it is recommended to use the default of 2. In the diagram the depth is the numbers of Max Pool or Deconvolution operations.

The **Number of Features in the First Layer** corresponds to the last shape specification coordinate in the diagram in the first layer on the left, and it determines how many different types of features (edges, corners, blobs...) the network will learn to detect in the first layer. In the diagram, after the first convolution, 16 features are considered. Thus, the number of features in the first layer is 16.

In most cases the default of 16, which is the minimal value, delivers good results for testing the training settings. Once it is ensured, the trained network performs well, larger values can be used for further improving the performance.

The subsequent layers in the contracting branch will double the number of features in every step. This is a standard construction for these types of networks. In the contracting branch the image resolution is reduced while the information depth (number of features) is increased. Thus, increasing the parameter also increases the amount of GPU memory required.

Control the **Size of** the **Convolutional Kernel.** A kernel is a feature detector, i.e., a n x n x n voxels box with a defined pattern as described here 9. The size n is the number of voxels in one direction and must be an odd number. Common values are 3 or 5.

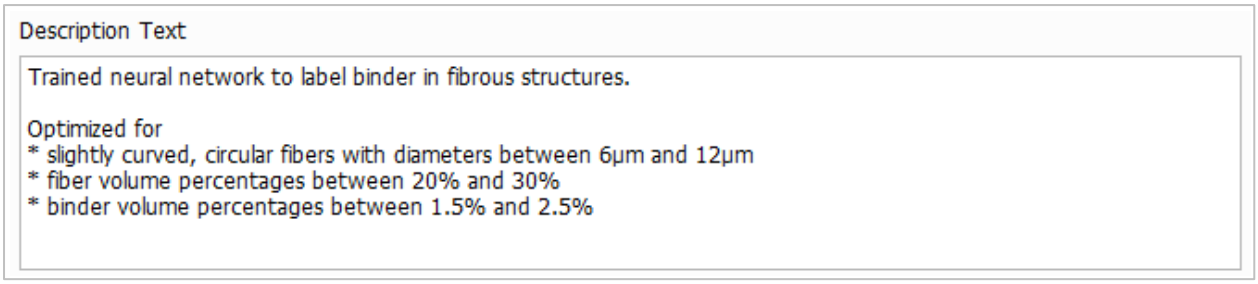
Optimizer

For the **Optimizer** select between **Adam** and **SGD**. The optimization algorithm iteratively updates the network weights 14 according to the current training data.

For many applications, the **Adam** optimizer leads to faster convergence, but the **SGD** (stochastic gradient descent) optimizer can result in a slightly better result. In most cases using **Adam** is recommended. If all other parameters work well to the current training, but the result is not already perfect, run a final production training run with **SGD**. For theory about Adam refer to Kingma and Ba, 2016 99, and for more information about the SGD method see Ruder, 2016 99.

Description Text

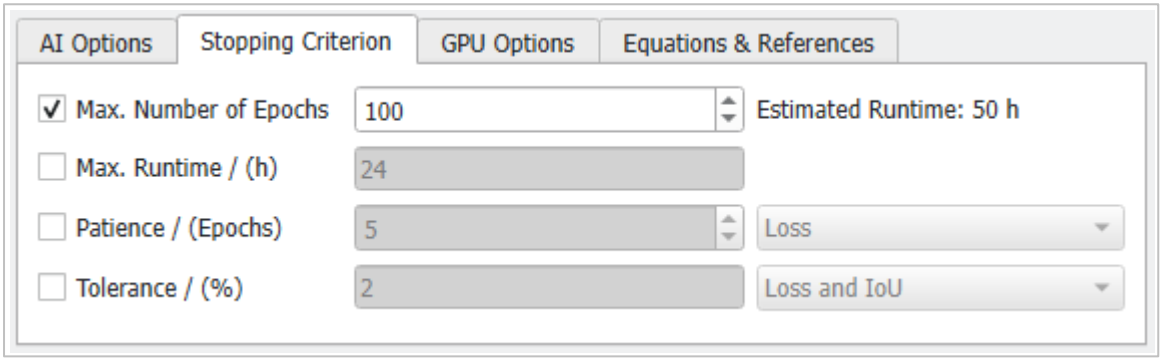
The **Description Text** can be edited as desired and will be displayed in the Apply Neural Network 61 dialog when loading the trained network. It should contain the purpose of the model and the assumptions made during training, e.g., the fiber diameter range and the solid volume percentage range. Thus, the user applying the model can decide if the model applies to their structure.



Important! The description cannot be changed after the training. Thus, always remember to enter a meaningful description before starting the training.

Stopping Criterion

Define the stopping criteria. If more than one criterion is selected, the training is stopped as soon one of them is reached.



Max. Number of Epochs

The max Number of Epochs determines how many passes over the data (epochs[51]) are made during training. The default of 100 takes some time except for the tiniest data sets. In most cases, smaller values can be used, but as long as the needed number of epochs for the current training is not known, run the training with 100 epochs and observe, when the training converges. Then, the training can be stopped manually[53]. Another possibility in this case is to use the stopping criteria Patience[49] or Tolerance[49].

If an Epoch Time[41] is entered under the **AI Options** tab, the **Estimated Runtime** of the training is shown on the right. If additionally the stopping criterion Max. Runtime[48] is selected below, the value on the right of the number of epochs is the minimum of the given maximum runtime and the value given by the number of epochs and the epoch time.

Max. Runtime / (h)

Specify a maximum runtime in hours for the training.

Patience / (Epochs)

The training is stopped if the chosen criterion did not change for the entered number of epochs. Choose Loss, Intersection over Union (IoU) or both from the pull-down menu. The loss is the validation loss shown in the progress dialog and IoU is the mean IoU. If both are selected, the criterion must be matched by both parameters.

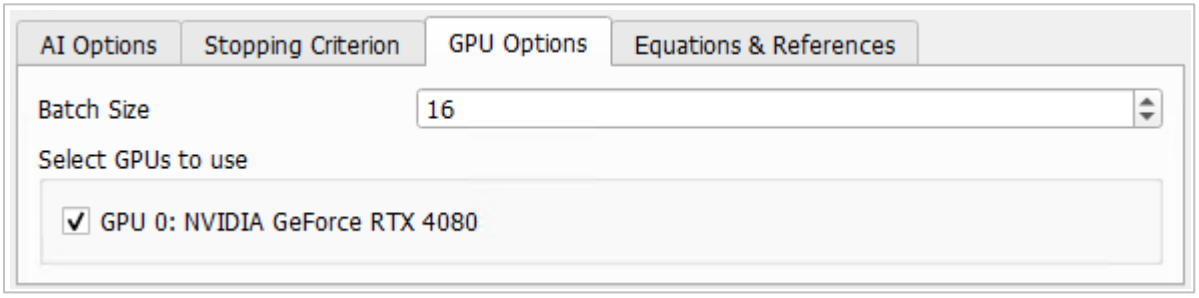
Tolerance / (%)

The training stops, after the entered tolerance for the chosen criterion is reached. The selectable criteria are Loss and Intersection over Union (IoU) and the combination of both. The loss is the validation loss shown in the progress dialog and IoU is the mean IoU. If both are selected, the criterion must be matched by both parameters.

The tolerance considers the relative change from the last to the current epoch. The tolerance has to be lower than the entered value for two epochs in a row to stop the training.

GPU Options

In the **GPU Setup** panel define the parameters for the used graphic card(s).



Batch Size

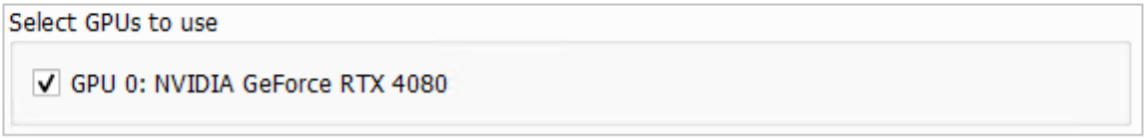
The **Batch Size** is the number of samples, which the training uses simultaneously. For example, with the default batch size 4, during training the error is computed, and the network updated for four windows at the same time, before moving on to the next four windows. A value of 1 slightly decreases training efficiency and can result in noisier training. For example, if a window contains an uncommon geometric configuration, the training could take a step in the wrong direction when computing the updated network weights for the next step. In literature there is usually a warning about too large values for batch size due to the potential of overfitting, but in 3D image analysis here the batch size is limited by the available GPU memory. If the value is chosen too high, a warning dialog appears, when starting the training. It is recommended to set the value as high as possible with the available GPU memory, so that the training works without

giving an error. This is often a value between 1 and 16 depending on the GPU hardware.

If multiple GPUs are selected, the batches are distributed to equal numbers on the different GPUs.

Select GPUS to use

If a suitable graphics card is detected during installation of GeoDict, the GPU mode is used for running the training, otherwise it is running in CPU mode.

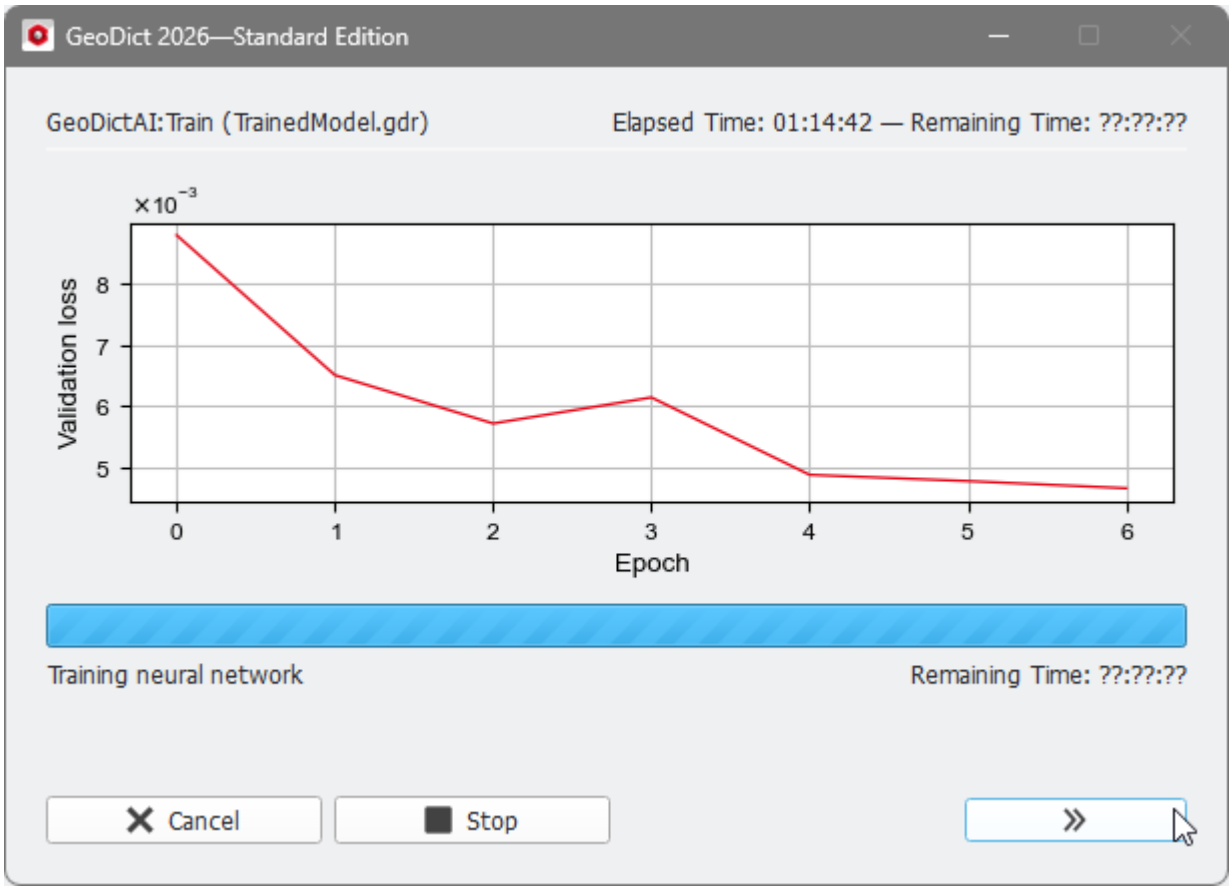


If multiple GPUs are available, select on which it should run. The training can also run on multiple GPUs. If selecting more GPUs than licensed, an error message appears when clicking Run.

1.5.2 Run Training

After setting the parameters click **OK** to close the dialog.

After clicking **Run** observe the training in the progress dialog and for more details open the **GeoDict Console** by clicking on the double arrow in the bottom right of the dialog.



For each epoch the number of windows[43] used for training and validation is given in the first two lines.

Afterwards find the IoUs[52] for each class.

The epoch[51] number, epoch duration, loss[52] and mean IoU[52] are found in the fourth line.

```
Saving model as D:/hilden/UserGuide/GeoDictAI/TrainedModel/Epoch_1.gnn.
New minimum loss found, copying to BestModel.gnn
---
n_windows reached, mode = train, windows_produced = 157855, windows_skipped = 0
n_windows reached, mode = test, windows_produced = 17539, windows_skipped = 0                    One Epoch
IoUs: [0.977311415988459, 0.8612367360262916] (Mean: 0.9192740760073753)
epoch 2: duration 700.5, training loss: 0.006480223041991352 5, validation loss: 0.005912627052548434, validation IoU: 0.91
tolerances loss: 0.06878051458999981 tolerance IoU: 0.006893524213611841
Saving model as D:/hilden/UserGuide/GeoDictAI/TrainedModel/Epoch_2.gnn.
New minimum loss found, copying to BestModel.gnn
---
n_windows reached, mode = train, windows_produced = 157855, windows_skipped = 0
n_windows reached, mode = test, windows_produced = 17539, windows_skipped = 0
IoUs: [0.977228292259437, 0.8577944056440158] (Mean: 0.9175113489517264)
epoch 3: duration 744.4, training loss: 0.005903617846092047, validation loss: 0.005993468302500226, validation IoU: 0.917
tolerances loss: 0.013672644872290 54 tolerance IoU: 0.0019175206846960048
Saving model as D:/hilden/UserGuide/GeoDictAI/TrainedModel/Epoch_3.gnn.
---
```

Epoch

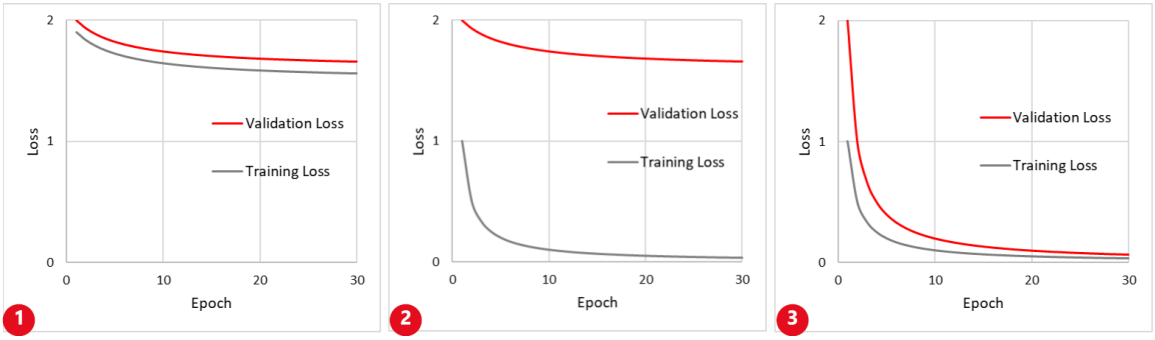
An epoch is a single pass over all the data. After each epoch, the current values for the neural network performance, i.e., the **IoU**, **training loss** and **validation loss,** are shown in the console and a .gnn file is written to the result folder for every step. Also, observe the **duration** time of one epoch. If one epoch only needs a few seconds, the epoch time or the downsampling factor[41] should be increased.

Training Loss and Validation Loss

To consider the convergence during training, observe **Training Loss** and **Validation Loss** in the **GeoDict Console**. These correspond to the two subsets of the data defined by the Train / Test Split Factor 42 set up in the **Train Neural Network Options** dialog. In neural network training the term "Loss" defines the current error of the neural network. The **Training Loss** is computed during each epoch while training on the training subset of data. The **Validation Loss** is computed while applying the current network of the epoch on the validation subset of data.

In the progress plot on the left of the progress dialog, the convergence behavior is visualized for the **Validation Loss**.

Ideally, both loss values converge towards zero, meaning zero error. In the following the three different scenarios that can occur in training are discussed:



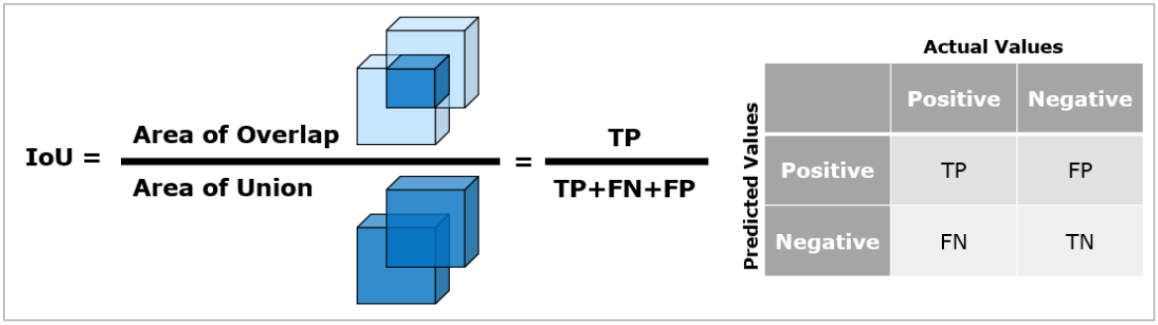
1. Both losses do not go down/stagnate: Either the network does not have enough context in a window, and the Window Size 43 must be increased, or the nature of the problem is too complex for the network and the Depth 45 or the Number of Features in First Layer 45 must be increased.
2. Training loss goes down, validation loss does not: This is usually a sign of overfitting, e.g., the network "memorizes" the training data and does not generalize to the test data. The solution is usually to feed in more data, e.g., by generating more training structures or training images.
3. Both losses go down and converge towards zero: The training converges and thus, the network works well on the training data and generalizes to the test data.

As the scale of the figures vary with the parameters in structure generation, it is currently not possible to specify a "good" numerical value for the loss. Consider the shape of the convergence for the current application and remember the values for similar trainings in the future to estimate the training performance. As described in the categorization above, the **Validation Loss** is the more important number to consider primarily.

IoU

The **IoU** (Intersection over Union) is a common tool to measure the current

performance of the neural network in each epoch. IoU considers the ground truth and the predicted data. The ground truth is the data taken from the output.gdt files, while the predicted data means the input data labeled by the current network. The score then is the ratio of the intersection and the union of the two. This can also be described with the confusion matrix[93].



A IoU score of 1 would be a perfect fit, meaning that all voxels were labeled correctly leading to FN = FP = 0. Thus, a value near 1 already indicates a very good model.

Similar to the **Validation Loss**, the IoU is measured on the test subset of the training data defined by the **Train/Test Split Factor**. But while the validation loss considers the confidence of the neural network, the IoU only considers the result (target material or not). Thus, the IoU is a more comparable measurement tool.

The IoU is computed separately for each material class to identify. They are shown in the third line of each epoch, followed by their mean value. Observe the mean value to learn about the overall training performance and the separate IoUs to find out if one material class is more difficult to identify as the others.

Know how! The IoU only makes sense for classification tasks as identifying materials or individual fibers or segmenting grayscale images. For enhancing grayscale images, always consider the validation loss[52].

Cancel or Stop Training

The training is run until the **Maximal Number of Epochs** is reached or the training is stopped manually. It is recommended to click **Stop** rather than **Cancel**, as far as **Stop** still produces a result file and keeps all results already generated and saved in the result folder. Contrary, **Cancel** deletes the complete result folder. It can be necessary to stop the training manually, if for example a too large number of epochs was selected, and the training already converges long before the number is reached.

1.5.3 Results

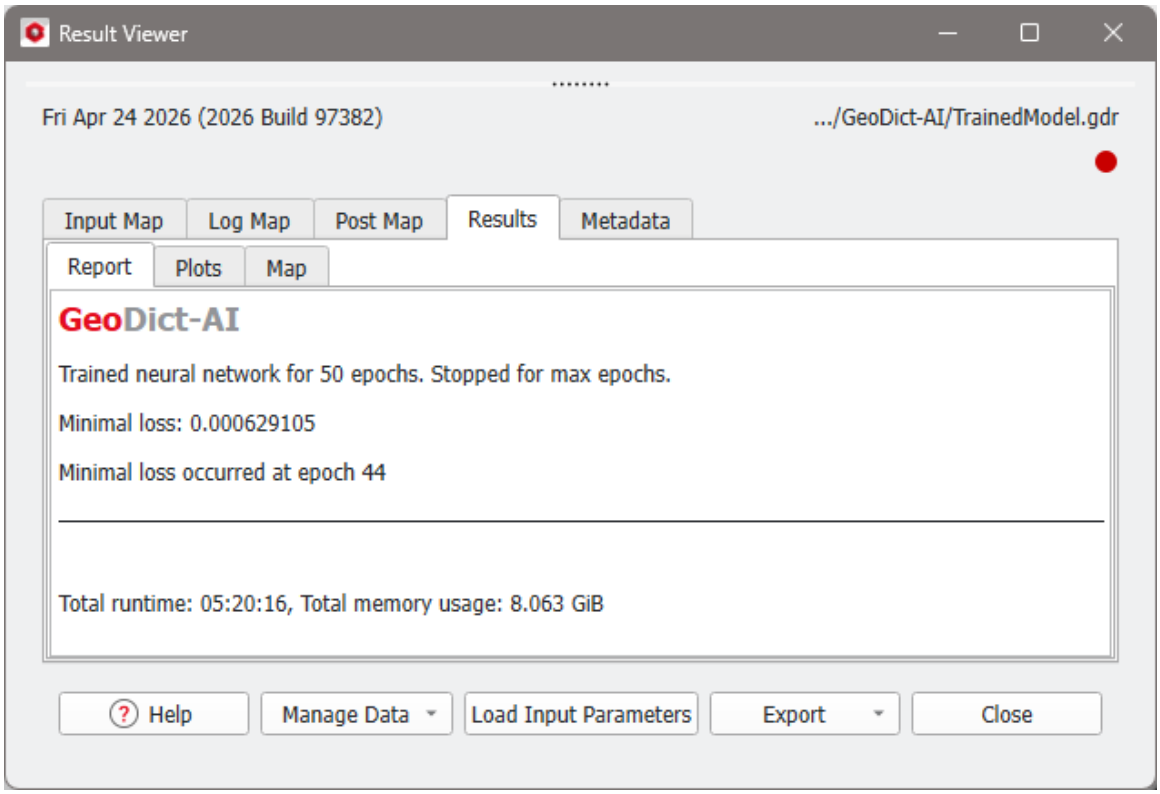
The training produces a GeoDict result file (*.gdr) and a folder with the same name. Both are saved in the chosen project folder (**File** → **Choose Project Folder...** in the menu bar).

Report

The GeoDict Result Viewer opens for the result file and statistical properties of the training are shown in the **Results – Report** subtab.

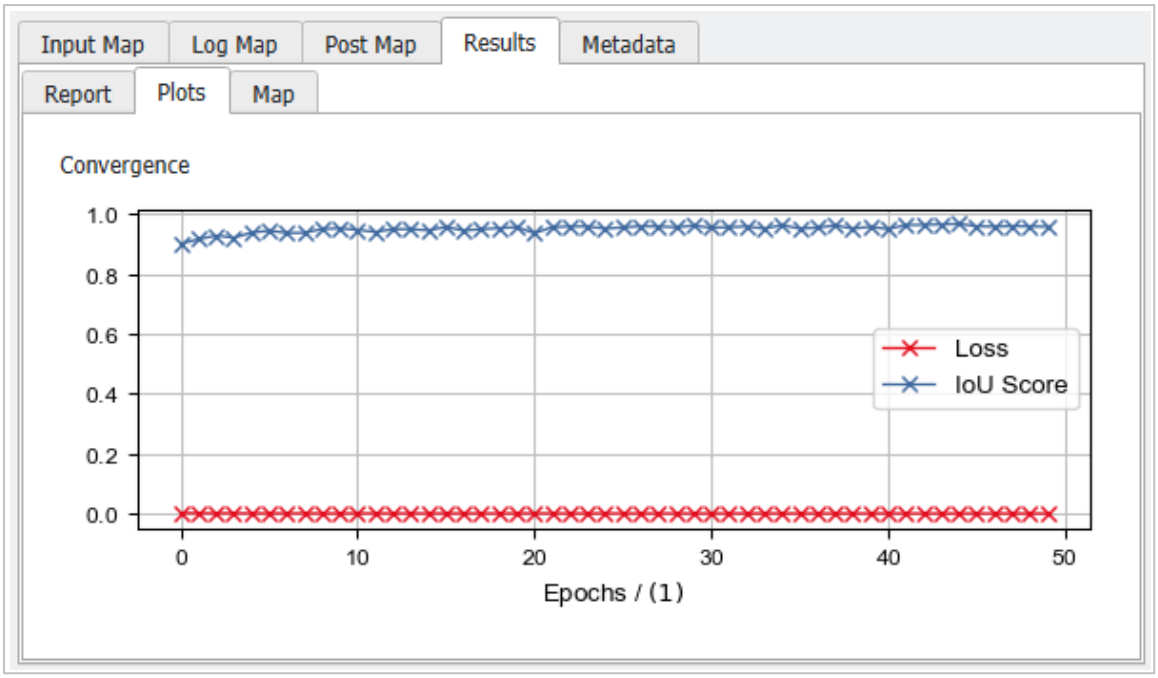
The first line shows after how many epochs the training was stopped and which criterion[48] led to the stop.

The **Minimal Loss** is the smallest Validation Loss[52], that occurred during the training. The epoch corresponding to this loss value is given in the last line of the report. While the training in the example was done for 50 epochs (epoch 0-49), the minimal loss (0.0028) occurred already at epoch 46. Thus, in this example the BestModel.gnn is not the last model Epoch_49.gnn, but Epoch_46.gnn.

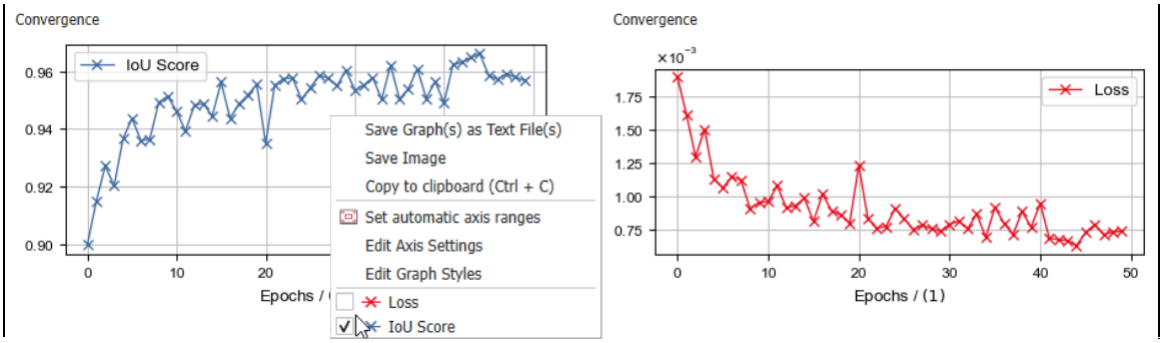


Plots

In the Result Viewer, under the **Results - Plots** subtab, find the **Convergence** plot showing the Validation Loss[52] and the mean IoU Score[52] for each **Epoch**.

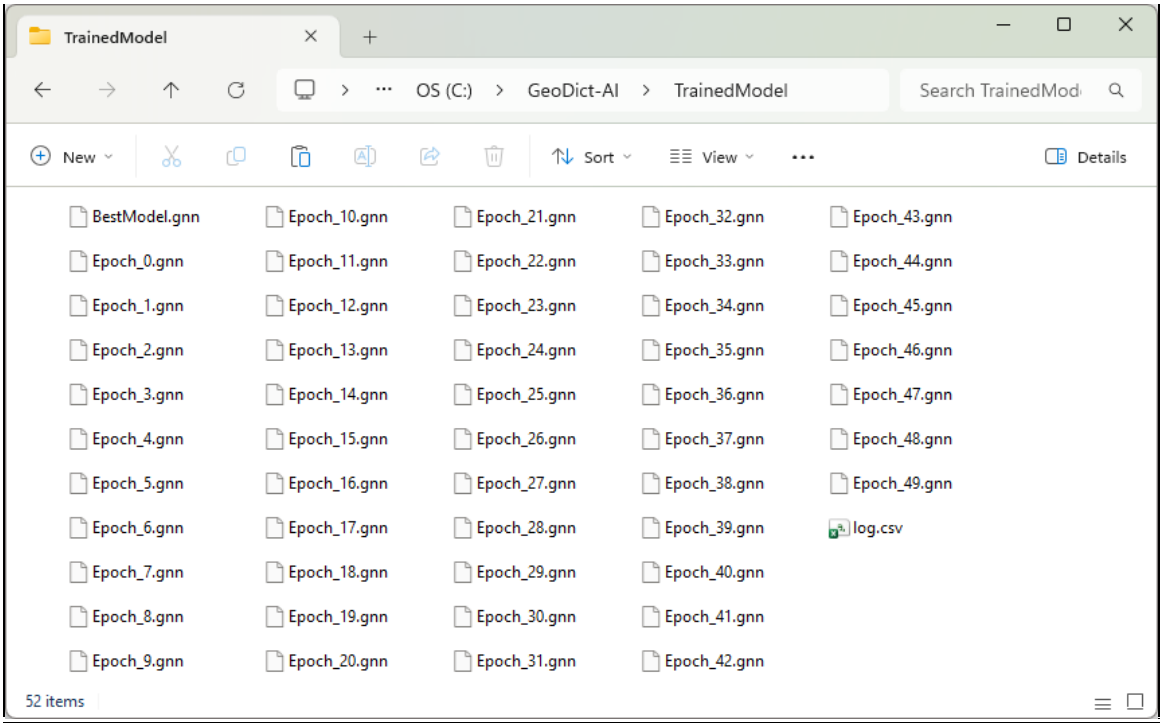


Right-clicking in the plot gives access to many post-processing options regarding the plot. In detail they are described in the Result Viewer topic. For example, select which plots should be shown.



Result Folder

Often the latest neural network performs best, but if the training convergence is not monotonous, this can be different. Thus, the result folder contains a **GeoDict Neural Network** (*.**gnn**) for each epoch and the neural network with the minimal validation loss, which is called the **BestModel.gnn**. Find the corresponding epoch in the Result Viewer as shown below 54. During the training in every epoch, when a new minimal loss is found, the corresponding *.gnn file is copied and renamed to BestModel.gnn.



The file **log.csv** documents the duration[51], loss[52], and the different IoU[52] for each epoch. These values are also shown in the progress dialog as explained here[50]. Inspect the file with a text editor. Here, it was opened with Excel.

	A	B	C	D	E	F	G
1	Epoch	Duration	Train Loss	Validation Loss	Mean IoU	Validation IoU (Class 1)	Validation IoU (Class 2)
2	0	382.0089	0.003612384	0.001889495	0.899963175	0.993105669	0.806820682
3	1	383.0903	0.00201138	0.001609235	0.914852569	0.993925517	0.835779621
4	2	386.764	0.001605473	0.001294493	0.927129171	0.994997444	0.859260898
5	3	383.279	0.0014335	0.001499887	0.920287031	0.994211185	0.846362876
6	4	382.6388	0.001319758	0.001130287	0.936569102	0.995525621	0.877612583
7	5	383.3898	0.00122813	0.001060384	0.943601488	0.996109501	0.891093474
8	6	383.7646	0.001207346	0.00114183	0.93594824	0.995576057	0.876320423
9	7	384.2219	0.001142385	0.001118883	0.936478053	0.995688468	0.877267637
10	8	383.6485	0.00114643	0.000906001	0.949101207	0.99655938	0.901643034
11	9	382.4574	0.00107156	0.000951118	0.951311563	0.99662708	0.905996047
12	10	383.2158	0.00104637	0.000957004	0.946144432	0.996338986	0.895949877
13	11	384.1156	0.001039865	0.001076368	0.939175935	0.995892965	0.882458906
14	12	384.4434	0.000967036	0.000911681	0.948234434	0.996537096	0.899931772
15	13	383.1079	0.000991761	0.000926582	0.948737627	0.99658464	0.900890614
16	14	384.2622	0.000986685	0.000983882	0.94442814	0.996131787	0.892724492
17	15	384.0162	0.000935987	0.000809171	0.956191241	0.997021981	0.915360502

Use neural network

As soon the training is finished, the resulting network **BestModel.gnn** and the intermediate Epoch_#.gnn networks from the result folder can be used.

Networks trained to identify fibers or distinguish between material phases can be validated with Validate Performance[84].

If the network is trained to distinguish between material phases, it can be applied with the Apply Neural Network 58 option.

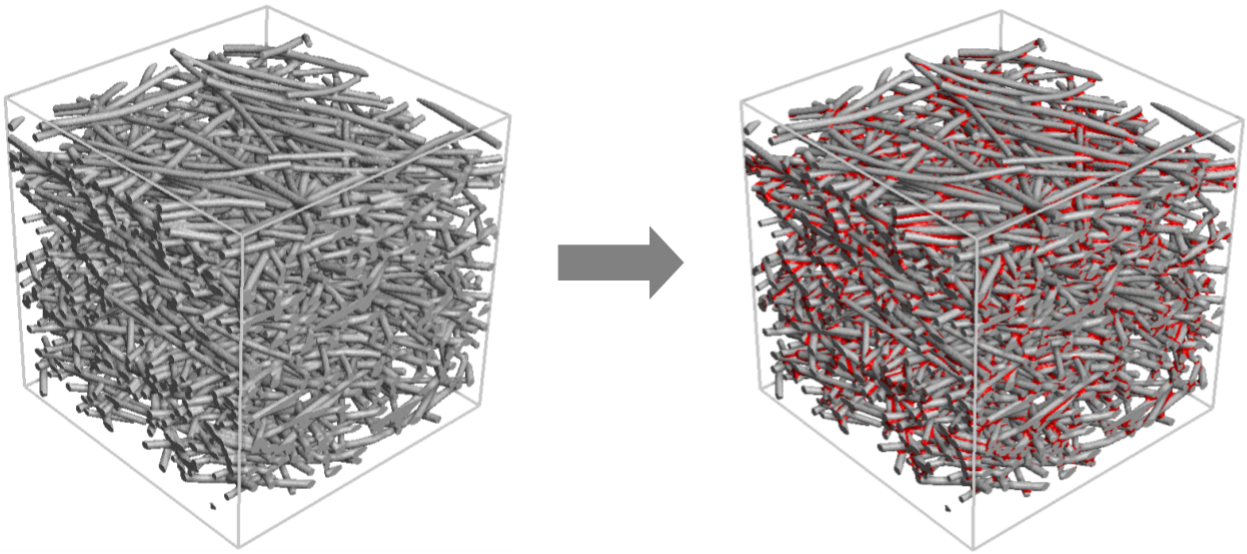
If networks are trained to enhance 3D-scans, they can be applied with Enhance Grayscale Image 71.

Networks trained to segment grayscale images can be used with Segment Grayscale Image 77.

Otherwise, if networks are trained to identify individual fibers, use **FiberFind Identify Fibers (AI)** as described in the FiberFind chapter.

1.6 Apply Neural Network

After producing training data 33 and training 39 neural networks, the networks can be applied to all structures with statistical parameters fitting to the network.



Note! Use GeoDict-AI **Apply Neural Network**, to apply neural networks trained to distinguish between material phases.

If the network is trained to identify fibers, use FiberFind **Identify Fibers (AI)**. Refer to the FiberFind chapter for more information.

If a network is trained for grayscale images use Enhance Grayscale Image 71 or Segment Grayscale Image 77 depending on the use-case.

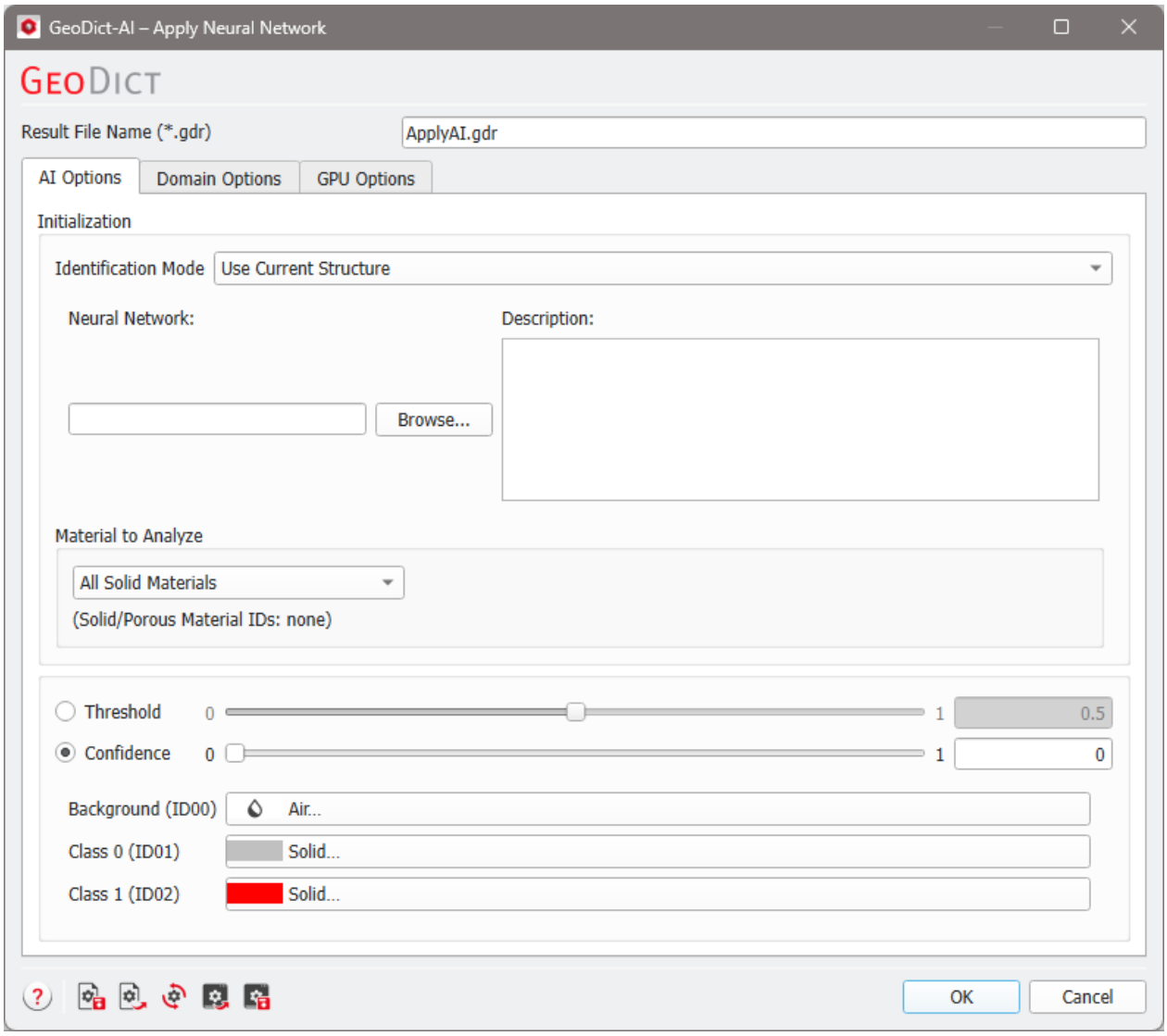
The Apply Neural Network command:

- Options 58
  Define the parameters for applying the neural network.
- Results 66
  Learn about the Apply Neural Network results.

1.6.1 Options

To start, load the structure to analyze and select **Apply Neural Network** from the pull-down menu in the GeoDict-AI section.

Click the **Options**' **Edit...** button.



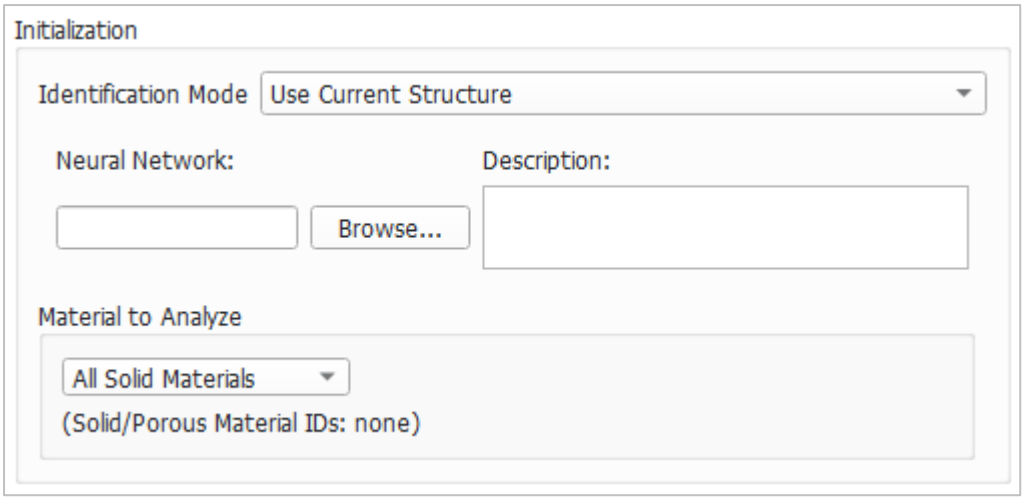
Tabs of Apply Neural Network:

- AI-Options 59
  Load the neural network that should be applied and select which materials to analyze.
- Domain Options 64
  Decide if periodicity should be applied.
- GPU Options 65
  Select a GPU and enter a batch size.

AI-Options

Set-up the material identification.

Initialization

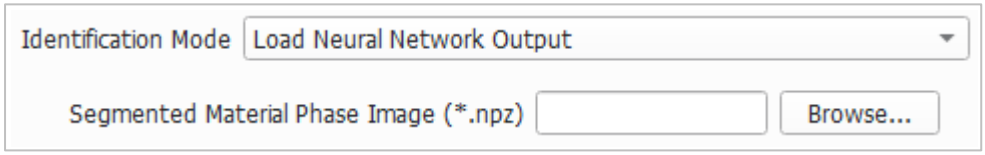


Identification Mode

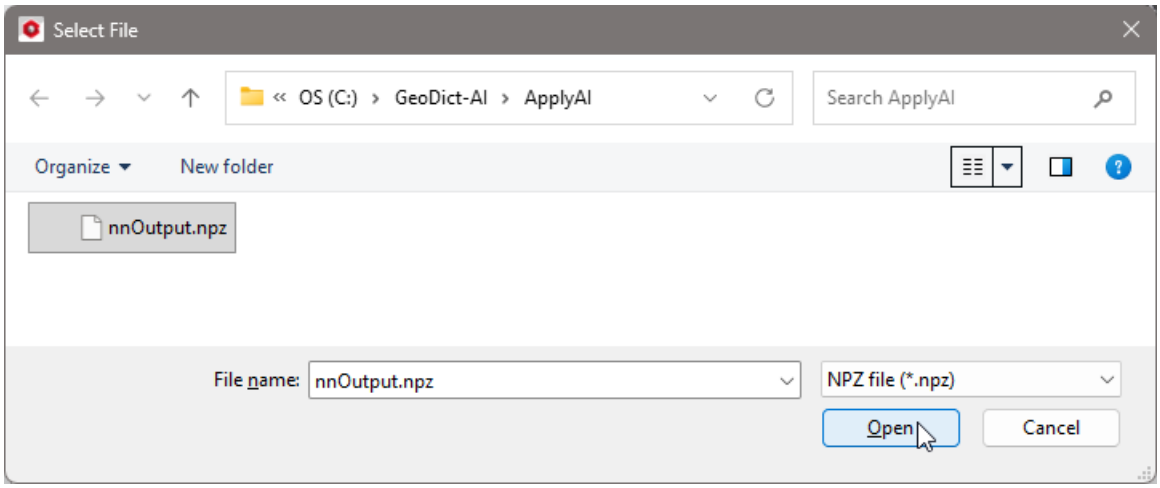
Two **Identification Mode** choices are available: **Use Current Structure** and **Load Neural Network Output**.

The default **Use Current Structure** applies GeoDict-AI to the structure in memory.

Occasionally, the **Load Neural Network Output** may be very useful, if different thresholds[63] or confidence[63] values should be tested, without having to run the identification for each voxel with the neural network again. Another application would be, that the identification was not finished, but an intermediate *.npz file is saved. This file can be loaded and thresholded to find out, if the identification is already good. A visualization of the confidence fields and the threshold is shown here[67].



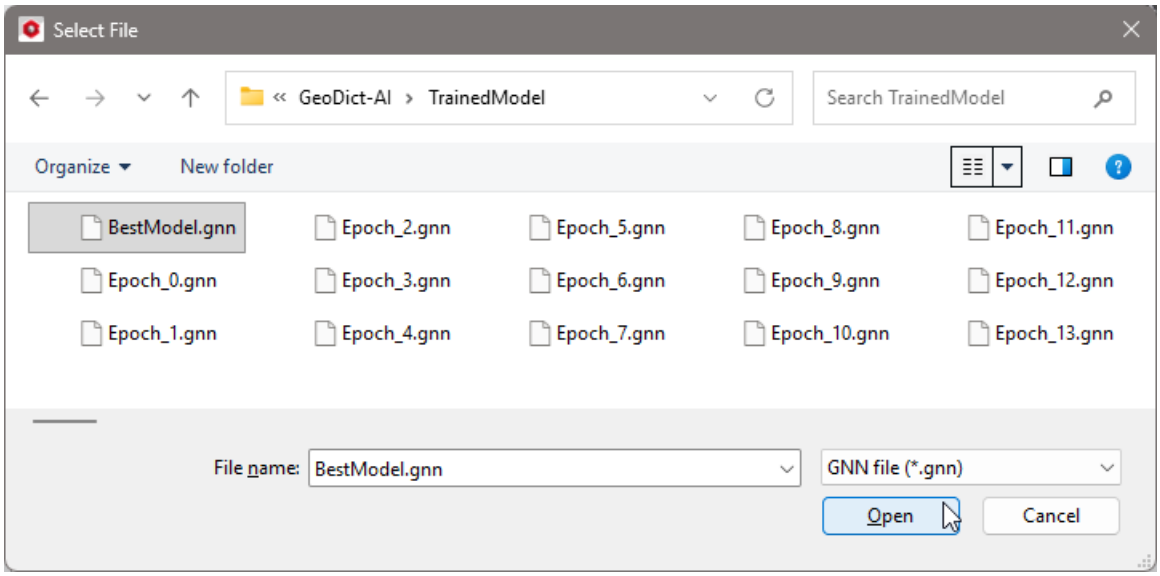
Browse to a GeoDict result folder of a previous run of GeoDict-AI **Apply Neural Network** and select a nnOutput.npz file from that folder. Accordingly, the Material to Analyze[61] and the Neural Network to be used cannot be changed, and the loaded structure needs to be the same.



Neural Network and Description

Browse for the neural network trained with **Train Neural Network** and most suitable for the structure under consideration. For this, under **Neural Network** click **Browse**.

Select a neural network *.gnn file, for example the **BestModel.gnn** from a prior training.



In the **Description**, the current constraints for the application of the neural network are listed, as entered in the Train Neural Network 47 dialog, e.g., the diameter range of the fibers the neural network was trained for.

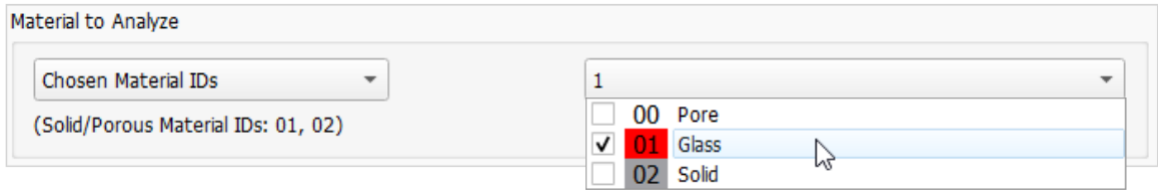
Material to Analyze

The **Material to Analyze** panel offers the choice of whether GeoDict-AI should be applied to all solids in the structure or only on a subset, defined either by choosing

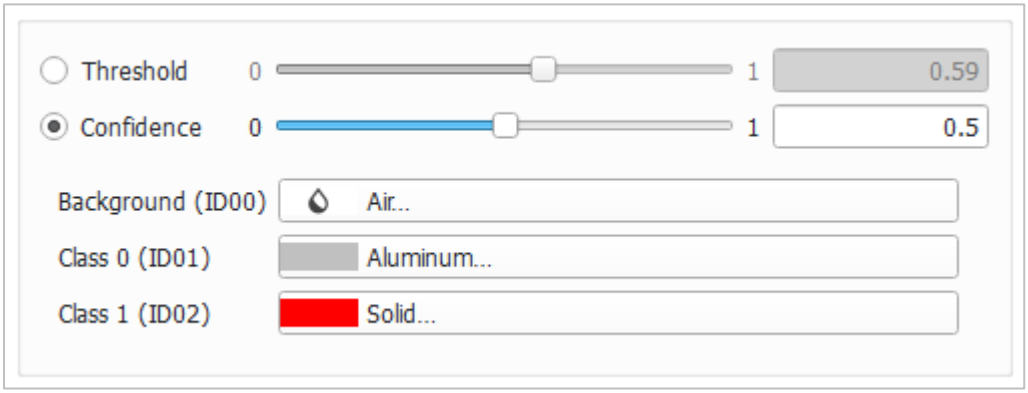
material IDs or materials.

If **Chosen Material** is selected, select the material to analyze from the pull-down menu on the right. All other materials will be considered as background by the neural network.

If **Chosen Material IDs** is selected, choose the material IDs to analyze from the pull-down menu on the right. All other material IDs will be considered as background by the neural network.



Transforming confidence field to structure



The neural network splits the material to analyze into different material classes. For each resulting class a confidence field is computed. These fields contain values between 0 and 1. A value of 1 means, the neural network is perfectly sure, that this voxel should be assigned to the corresponding class, while a value of 0 means the voxel definitely does not belong to the corresponding class.

Each voxel then is labeled according to the field with the highest value. For example, consider a structure where three materials should be identified and the field values for one voxel are as

follows:

- class 0 (cellulose fiber): 0.7
- class 1 (elliptical fiber): 0.25
- class 2 (binder): 0.05

In this case, the voxel is labeled as class 0 (cellulose fiber).

The parameters **Threshold** and **Confidence** are post-processing parameters. They can be used in combination with the Identification Mode 60 **Load Neural Network Output**. These parameters influence the labeling of voxels according to the probability values in the output fields (*.npz) from previous **Apply Neural Network** runs.

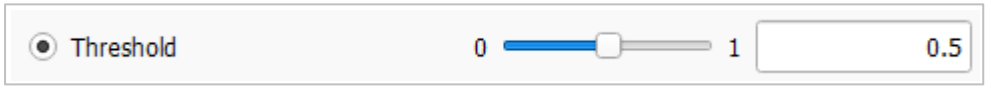
Use Threshold 63 for two-channel outputs, only, for example to prevent over-segmentation.

Use a higher Confidence 63 value if you want to find the voxels, where the network was uncertain.

Threshold

The **Threshold** allows proficient users to influence the decision to which material class a voxel belongs. The parameter is only available if the chosen network is trained to differentiate between exactly two material phases. For three or more material phases this parameter is grayed out.

If the identified features are over segmented, choosing a smaller threshold can lead to better results. For this, the confidence field (*.npz) can be loaded, without running the complete identification process again.



Find a visualization of the confidence field (*.npz) and threshold here 67.

For most applications, however, the default values can be kept because the default threshold of 0.5 usually produces good results for a well-trained network.

Important! This parameter only has an impact on two-channel results, i.e., distinguishing between only two material phases.

Confidence

Specify the minimum required confidence value for selecting a class for a voxel.

If a positive confidence value is specified and none of the fields has a higher probability value, the material is set to "undefined" in these uncertain locations, and material ID 255 is assigned. In the above 62 example, consider a confidence value of 0.8. In this case, the voxel would be labeled "undefined" instead of class 0.

The values of the confidence fields of all classes sum up to 1. Thus, it is impossible that

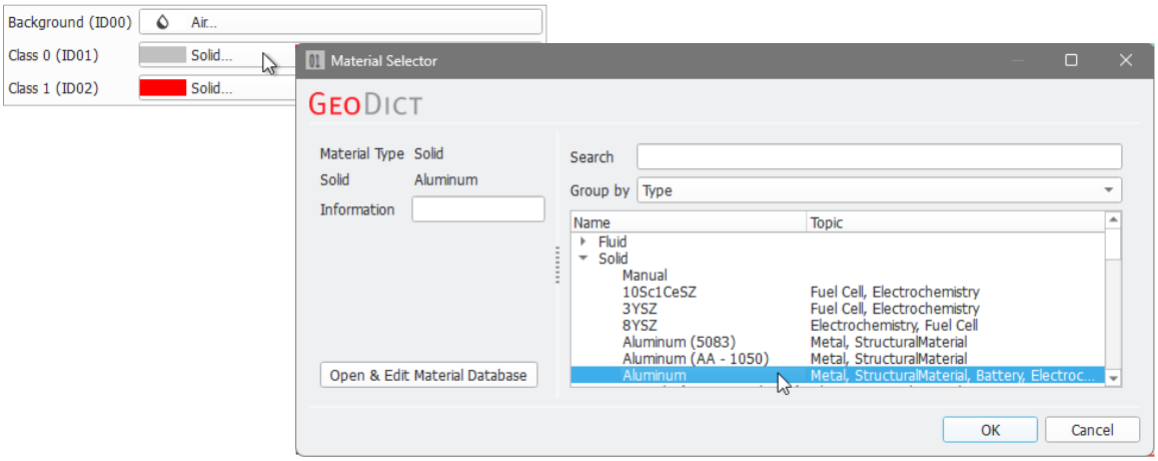
for any voxel all confidence values are smaller than 1/n, where n is the number of classes. This is why setting the confidence value below 1/n has no impact. Therefore, the default value is set to 0, meaning all voxels are labeled as one of the classes as described above 62 and no uncertain areas will remain.

An example is shown here 67.

Material Selection

For all constituent materials in the output structure select a material by clicking on the material buttons.

The **Material Selector** gives access to selecting the desired material from the GeoDict Material Database. When none of the materials available in the database fits the preferred specifications, **Manual** should be chosen.



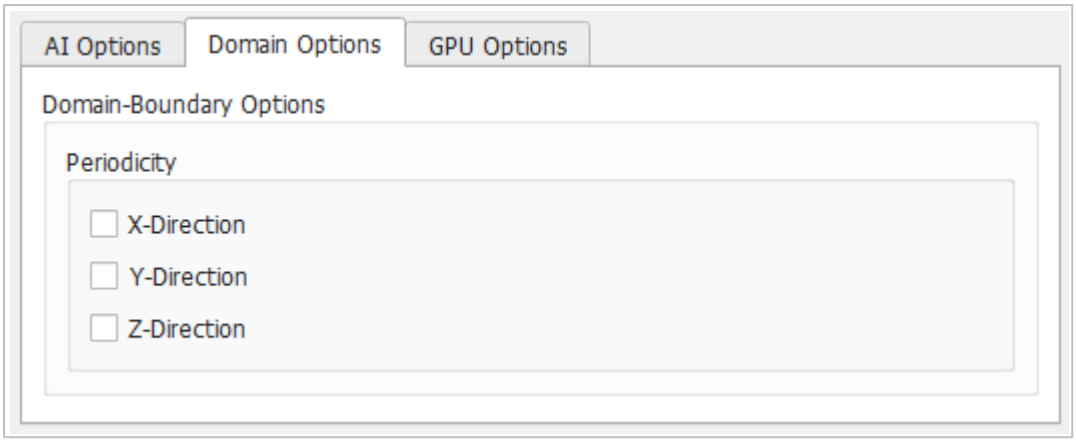
Alternatively, new materials can be defined in GeoDict's material database (Click **Edit Material Database**...).

The number of different **Classes** is defined by the neural network training and updates automatically. The counting starts with 0 for material ID 01. The **Background (ID00)** is ignored by the neural network and therefore is not counted as a class.

Domain Options

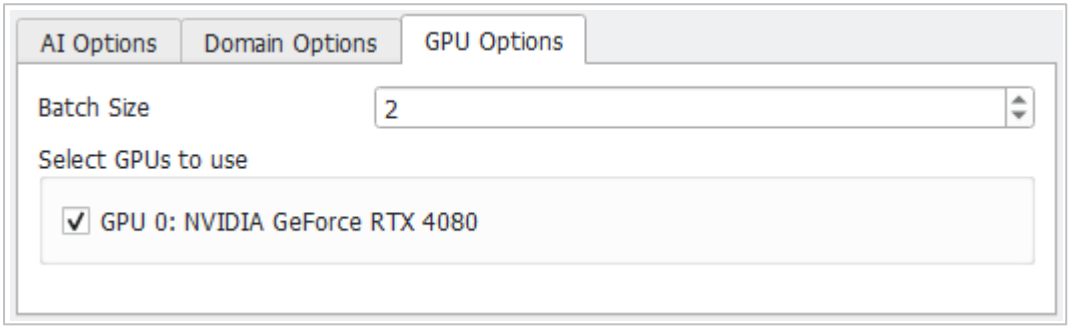
The underlying domain can bet set as periodic for the object identification by checking one or several of the boxes in the **Domain-Boundary Options** panel.

However, periodicity is unusual for most 3D scans and this setting is rarely changed.



GPU Options

In the **GPU Options** panel define the parameters for the used graphic card(s).



Batch Size

The **Batch Size** is the number of samples, which the neural network uses simultaneously. For example, with the default batch size 4, during applying the network the error is computed, and the result updated for four windows at the same time, before moving on to the next four windows. In literature there is usually a warning about too large values for batch size due to the potential of overfitting, but in 3D image analysis here the batch size is limited by the available GPU memory. If the value is chosen too high, a warning dialog appears, when starting to apply the neural network. It is recommended to set the value as high as possible with the available GPU memory, so that it works without giving an error. This is often a value between 1 and 16 depending on the GPU hardware.

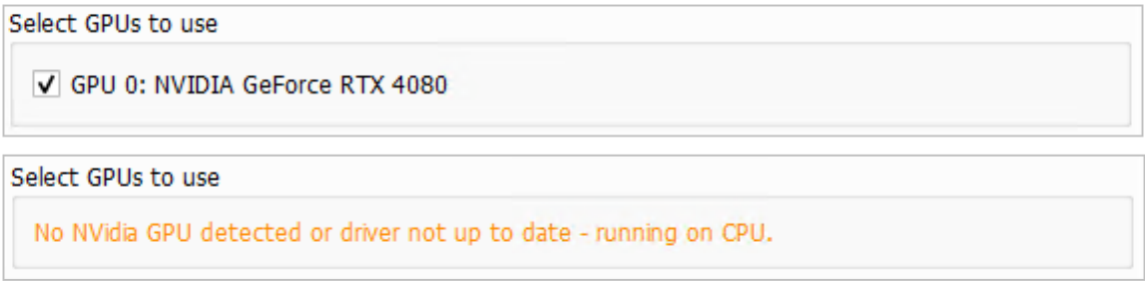
If multiple GPUs are selected, the batches are distributed to equal numbers on the different GPUs.

If the batch size was chosen too large for what is available on the graphics card, an error message appears suggesting to decrease the batch size.

Note! For CPU-based computation, the choice of batch size is not critical and can be left at the default.

Select GPUs to use

If a suitable graphics card is detected during installation of GeoDict, the GPU mode is used for applying the neural network, otherwise it is running in CPU mode.



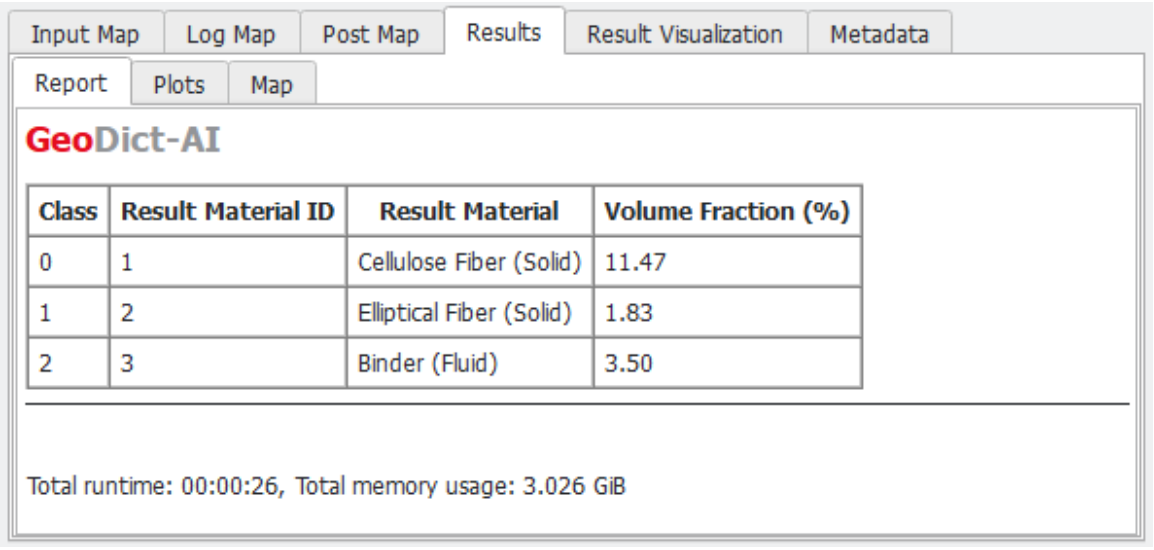
If multiple GPUs are available, select on which it should run. It can also run on multiple GPUs. If selecting more GPUs than licensed, an error message appears when clicking Run.

1.6.2 Results

After applying the neural network, a folder with the same name as the GeoDict result file is saved in the chosen project folder (**File** → **Choose Project Folder...** in the menu bar).

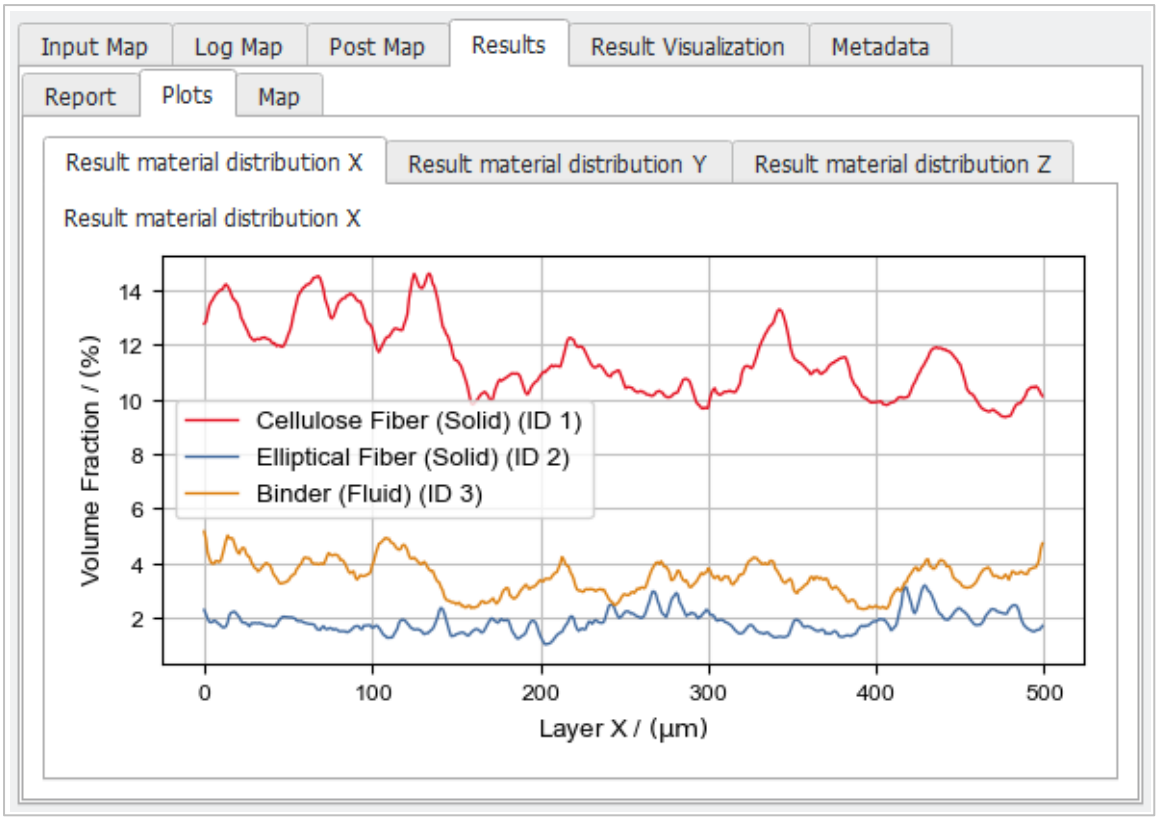
Report

The GeoDict **Result Viewer** opens for the result file. In the **Report** for each class identified in the structure the solid volume percentage is given.



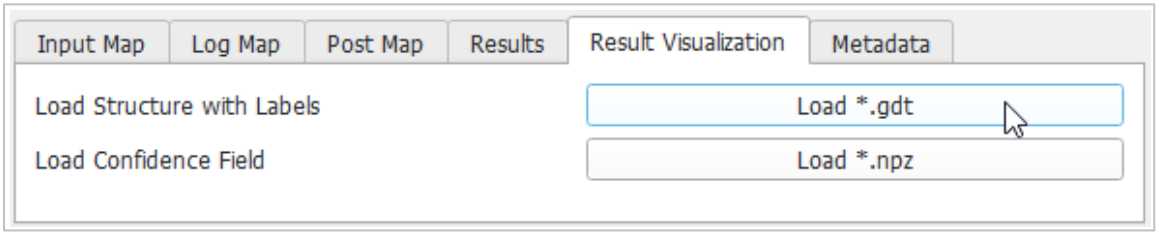
Plots

In the **Result Viewer**, under the **Results - Plots** subtab, find the volume percentage distribution of all classes per slice in X-, Y- and Z-direction.

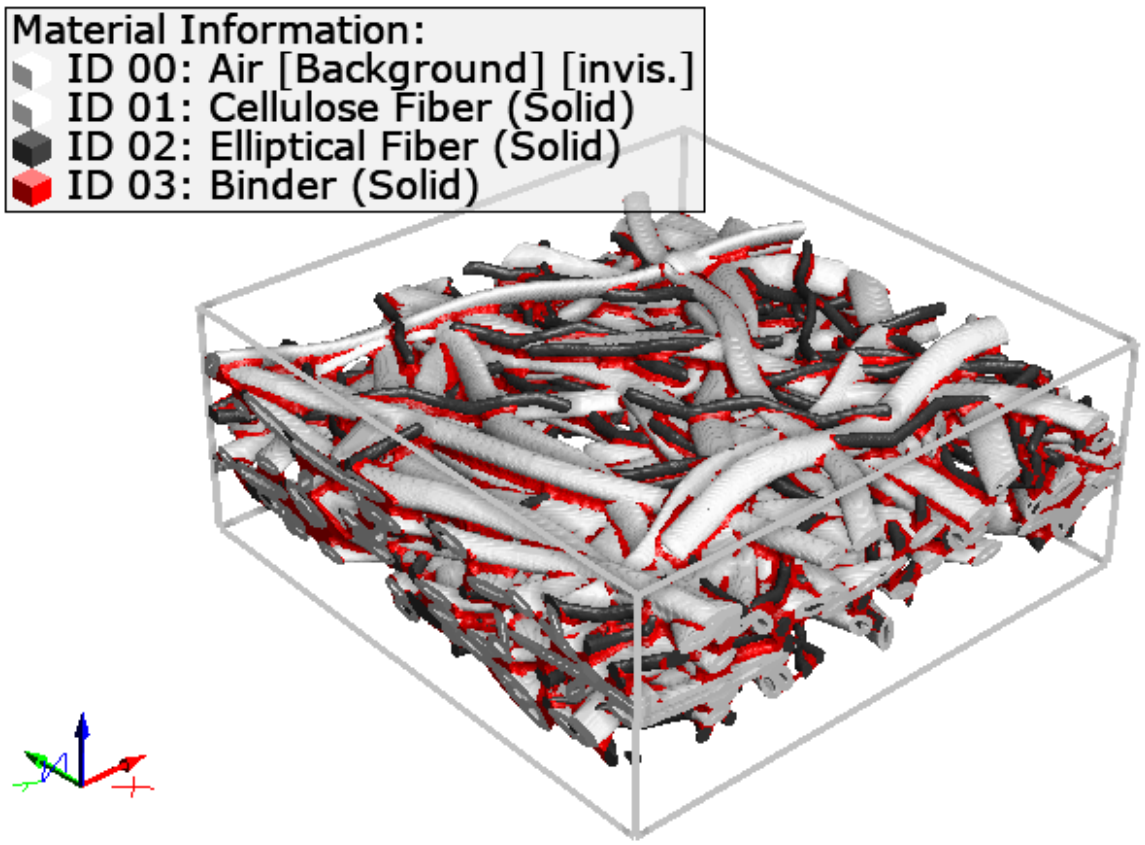


Result Visualization

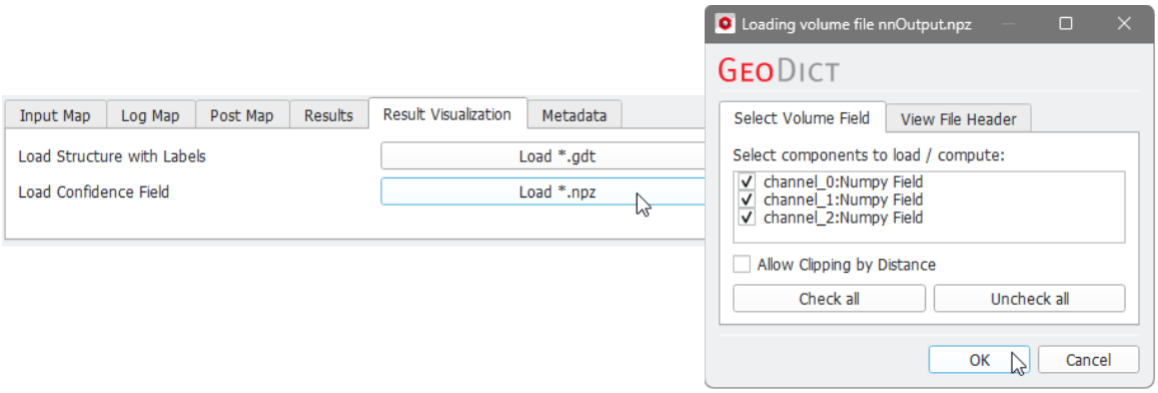
To visualize the target material identified by the network, move to the **Result Visualization** tab. There, click **Load *.gdt** for **Load Structure with Labels**.



The identified target materials are assigned to the next free materials IDs, and thus, are shown in the colors selected for these IDs.

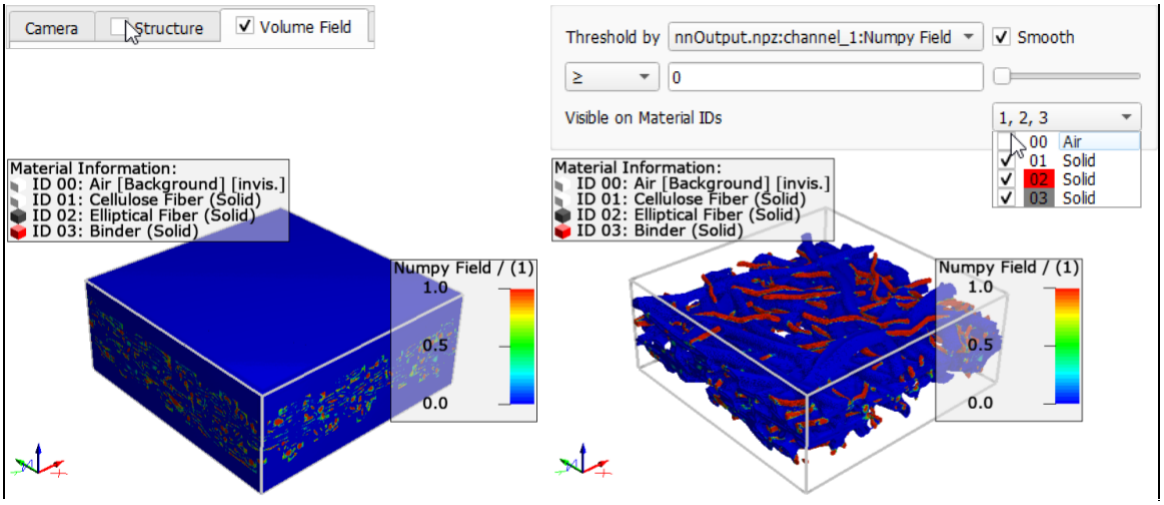


Click the **Load *.npz** button next to **Load Confidence Field** to load the unassigned data of the material identification. The analyzed material is split into the materials the network was trained for. For example, consider a neural network trained to distinguish the solid material into cellulose fibers (channel 0), elliptical fibers (channel 1) and binder (channel 2). Then each of them gets their own material ID after applying the network. For each channel a confidence field is created.

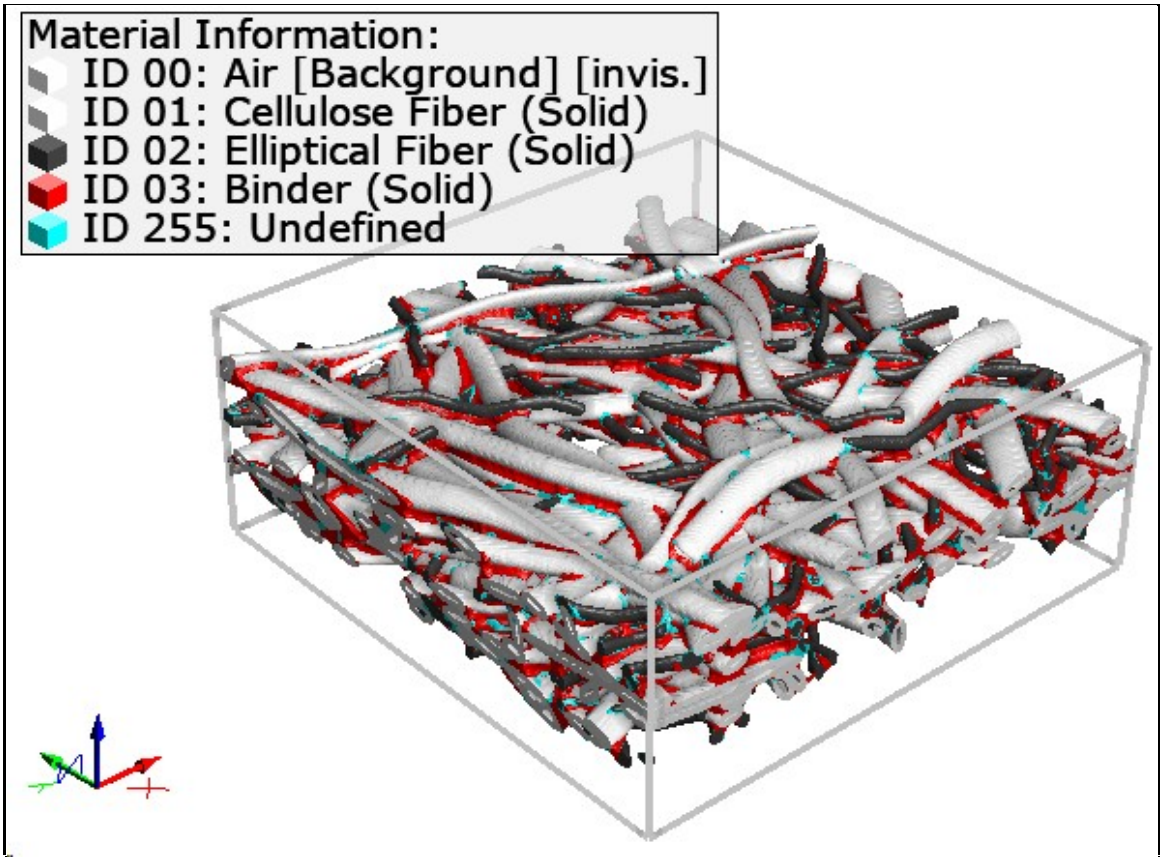


Set the structure to invisible, by unchecking the **Structure** tab in the **View Controls**, above the Visualization area. Move to the **Volume Field** tab to control the visualization of the confidence field. Deselect the IDs for **Visible on Material IDs** to only see the values of interest. In the example the confidence field for channel 1 (ellipsoidal fibers) is shown. The red voxels mean that the network was very confident that these voxels belong to channel one and blue means, it is very confident that these voxels do not

belong to circular fibers. Most voxels are either red or blue. Thus, the network was overall very confident with it's result.

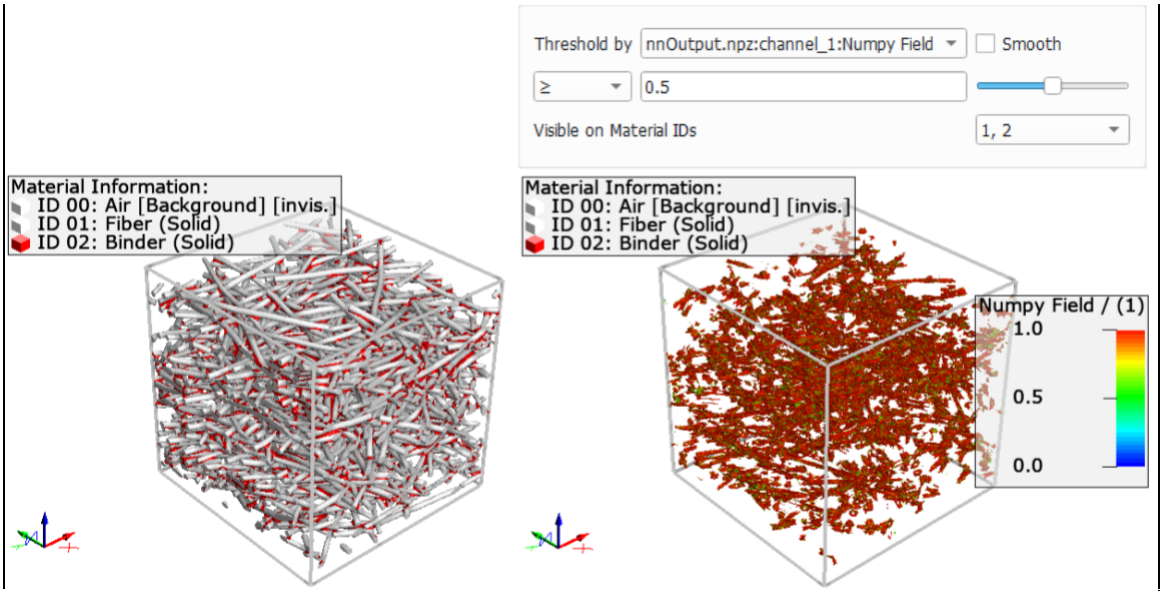


To show the voxels where the network was less confident you can use the confidence value as described here. In the following example the value was set to 0.9. Thus, all voxels assigned to ID 255 have a confidence value smaller than 0.9 for all three channels. For all other voxels the material is chosen depending on which channel has the highest confidence value.



You can also threshold the confidence field in the visualization area depending on the confidence value. For two-channel models, this is useful to visualize the **Threshold** for

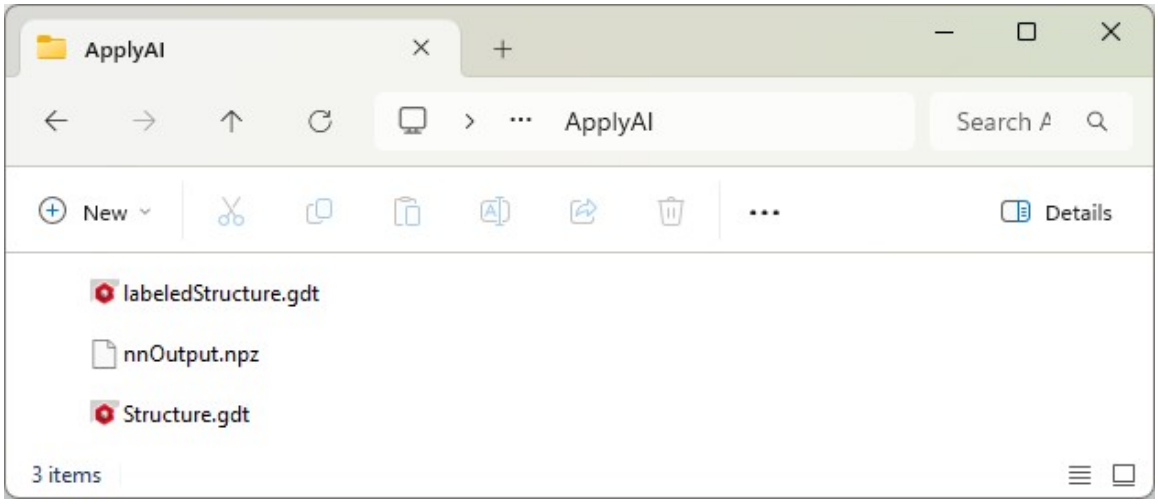
the two classes. For this, visualize the numpy field for channel 1 and change the clipping value to separate the values for the different materials. If the Threshold 63 was set to **0.5**, entering 0.5 for the volume field threshold and selecting **>=** as clip mode visualizes only the result values for the identified target material. For a well-performing network most of the values in the volume field should be near 1 (class 1) or near 0 (class 0), meaning the network is confident with the classification for the corresponding voxel. This can be observed in the example below. After clipping away all voxels with a value smaller than 0.5, most of the remaining voxels are red, i.e., near 1.



For more visualization options refer to the Visualization User Guide.

Result Folder

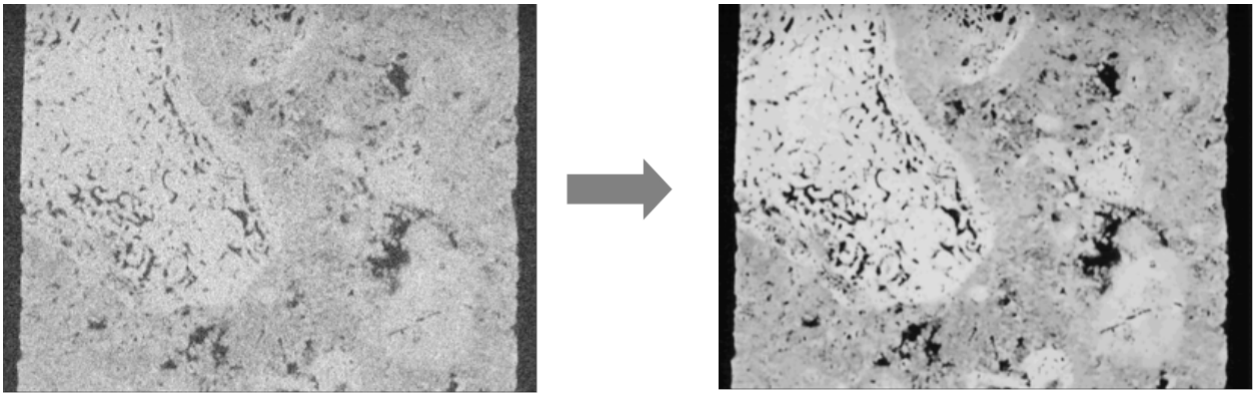
This folder contains the original structure model on which the identification was done (**Structure.gdt**) and the resulting structure with identified target material (**labeledStructure.gdt**). Also, the neural network output file (**nnOutput.npz**) is saved in the folder and can be reused as described here 60.



1.7 Enhance Grayscale Image

After producing training data[21] and training a neural network[39], the network can be applied to all images with statistical parameters fitting to the corresponding training data.

Use GeoDict-AI **Enhance Grayscale Image**, to apply neural networks trained to enhance 3D-scans.



The Enhance Grayscale Image command:

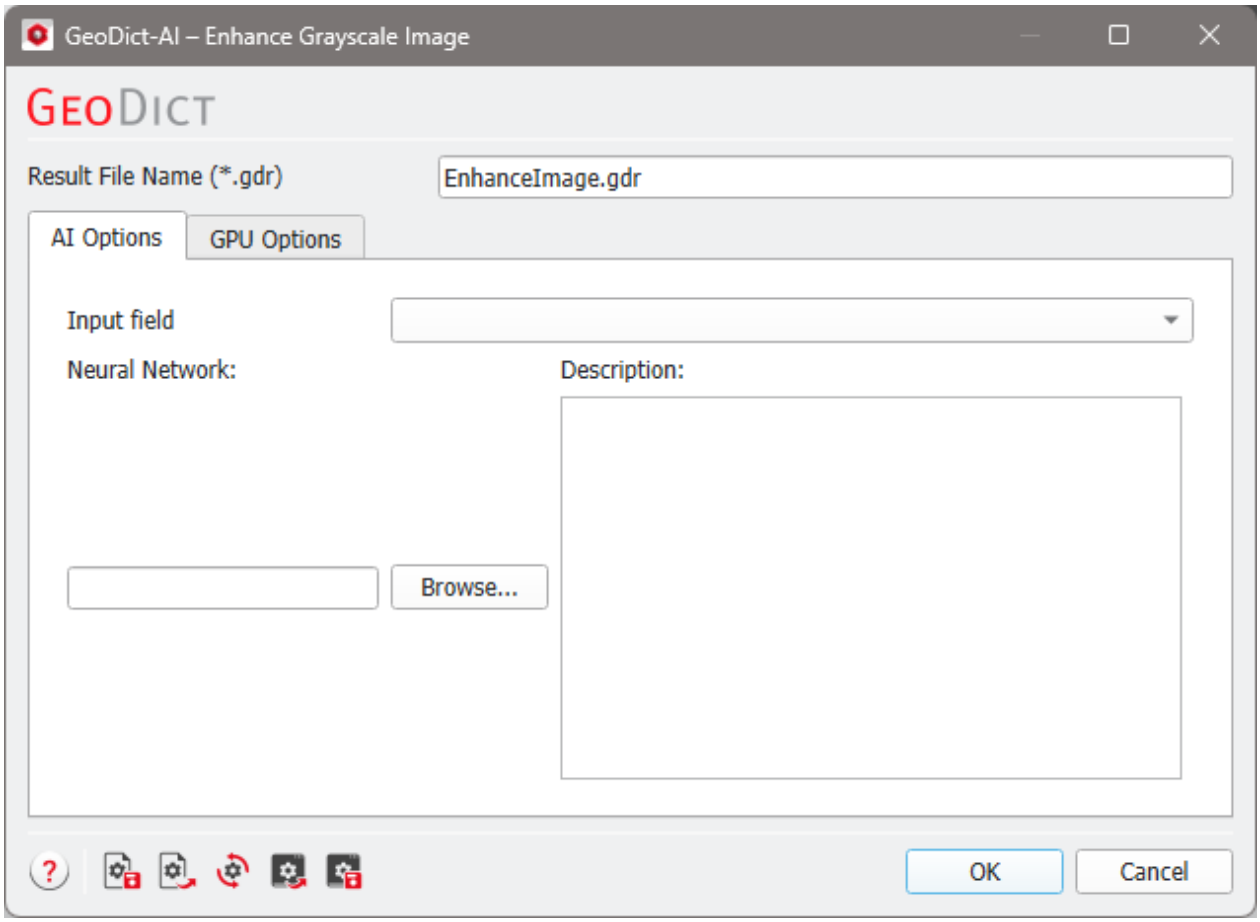
- Options[71]
  Define the parametersfor the image enhancement.
- Results[75]
  Learn about the Enhance Grayscal Image results.

1.7.1 Options

To start, load the image to analyze by selecting **File → Load Volume File ...** and browsing for the *.grw image to improve. Then, select **Enhance Grayscale Image** from the pull-down menu in the GeoDict-AI section. If the image is loaded with ImportGeo-Vol, use **Enhance Image (AI)** in the **Image Processing** dialog instead, as described in the ImportGeo-Vol User Guide. The image filter has the same functionality and the same parameters.

Click the **Options**' **Edit...** button.

The **Enhance Image (AI)** dialog is organized in two tabs: **AI Options** and **GPU Options**.

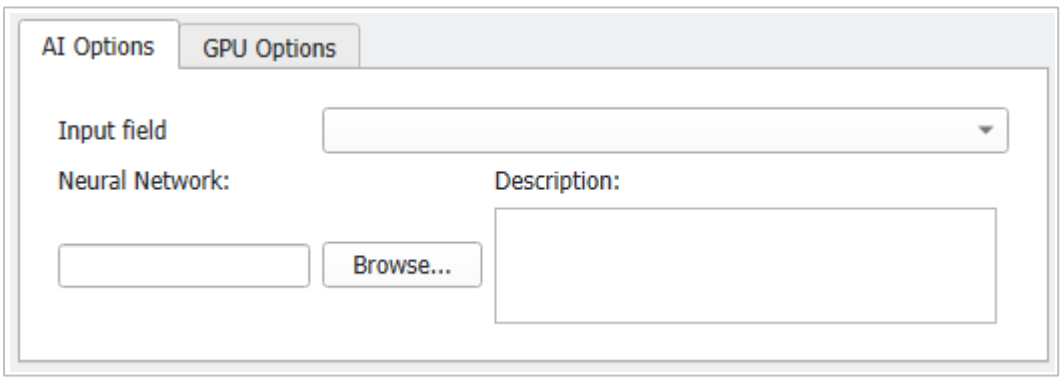


Tabs of Enhance Grayscale Image:

- AI Options 72
  Browse for the neural network to apply and select a field to analyze.
- GPU Options 73
  Select a GPU and enter a batch size.

AI Options

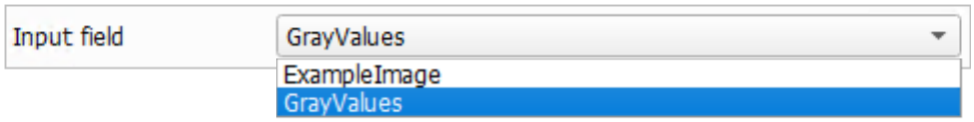
Define which loaded grayscale image should be enhanced using which neural network.



Input Field

Select one of the currently loaded grayscale images as **Input field** to enhance from the pull-down menu.

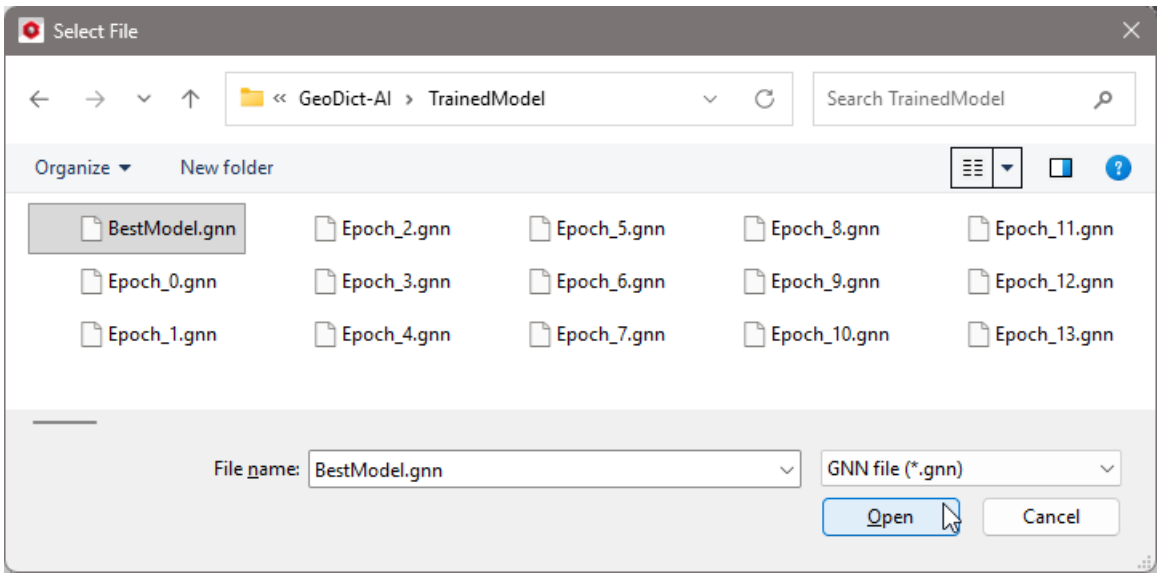
Several grayscale images can be loaded at the same time by checking **Keep existing Volume Fields** in the Loading volume file dialog, when loading the image with **File - Load Volume File.** Using ImportGeo-Vol also different volume fields can be loaded.



Neural Network and Description

Browse for the neural network trained with **Train Neural Network** and most suitable for the image under consideration. For this, under **Neural Network** click **Browse**.

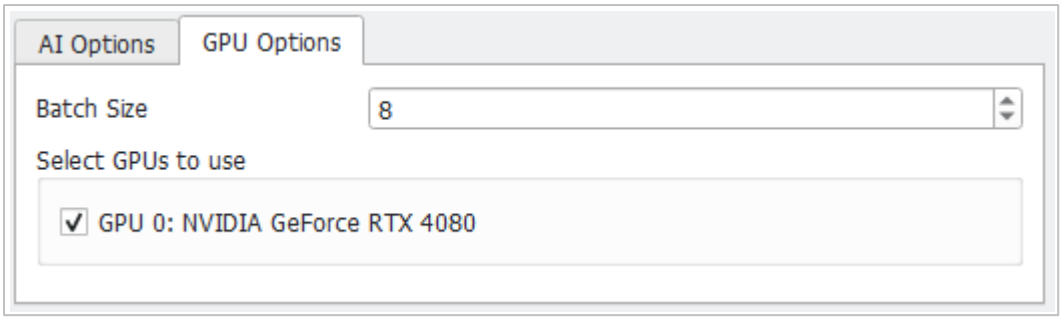
Select a neural network *.gnn file, for example the **BestModel.gnn** from a prior training.



In the **Description**, the current constraints for the application of the neural network are listed, as entered in the Train Neural Network 47 dialog, e.g., the diameter range of the fibers the neural network was trained for.

GPU Options

In the **GPU Options** panel define the parameters for the used graphic card(s).



Batch Size

The **Batch Size** is the number of samples, which the neural network uses simultaneously. For example, with the default batch size 4, during applying the network the error is computed, and the result updated for four windows at the same time, before moving on to the next four windows. In literature there is usually a warning about too large values for batch size due to the potential of overfitting, but in 3D image analysis here the batch size is limited by the available GPU memory. If the value is chosen too high, a warning dialog appears, when starting to apply the neural network. It is recommended to set the value as high as possible with the available GPU memory, so that it works without giving an error. This is often a value between 1 and 16 depending on the GPU hardware.

If multiple GPUs are selected, the batches are distributed to equal numbers on the different GPUs.

If the batch size was chosen too large for what is available on the graphics card, an error message appears suggesting to decrease the batch size.

Note! For CPU-based computation, the choice of batch size is not critical and can be left at the default.

Select GPUs to use

If a suitable graphics card is detected during installation of GeoDict, the GPU mode is used for applying the neural network, otherwise it is running in CPU mode.

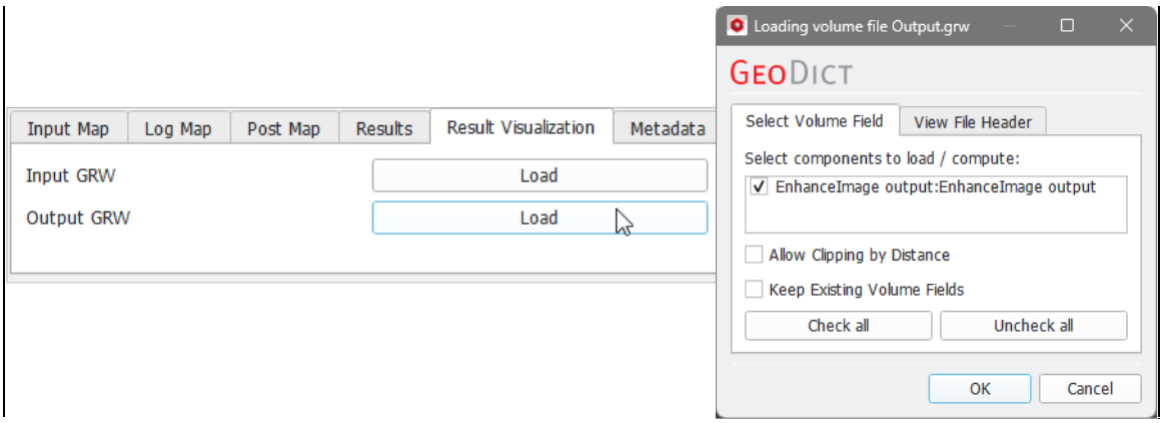
If multiple GPUs are available, select on which it should run. It can also run on multiple GPUs. If selecting more GPUs than licensed, an error message appears when clicking Run.

1.7.2 Results

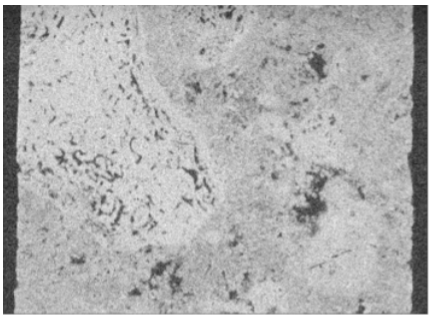
After applying the neural network, a folder with the same name as the GeoDict result file is saved in the chosen project folder (**File** → **Choose Project Folder...** in the menu bar).

Result Visualization

The GeoDict **Result Viewer** opens for the result file on the report, which only provides information on runtime and memory usage. From the **Result Visualization** tab, the input and output files can be loaded by clicking **Load**. If one of them is already loaded, check **Keep existing Volume Fields**. Then, you can switch between the two fields under the **Volume Field** tab.

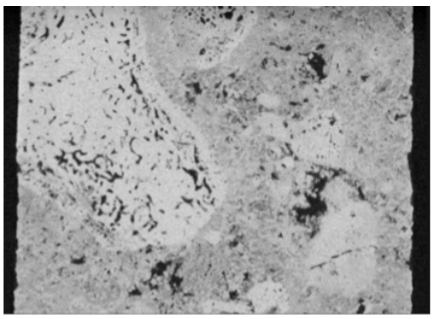


Compare the input and output images to validate the performance of the selected network.
In the example below, a carbonate sample was scanned once with short and once with long exposure time. The sample data was provided by [Petroleum Engineering Research Center](99).



Low quality scan

Short exposure time

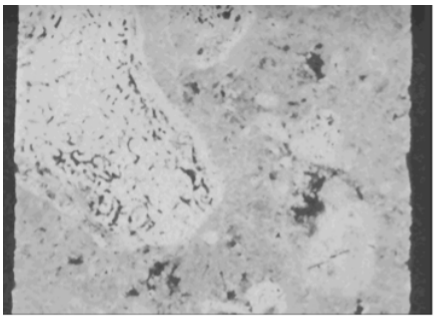


High quality scan

Long exposure time

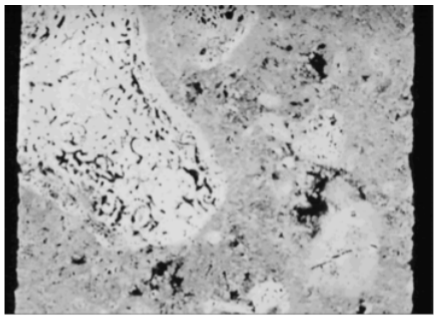
Using GeoDict -AI a neural network was trained to transform the low quality scan into the high quality scan.

Below find a comparison of applying the NLM-filter and applying the trained AI-model on the low quality scan, respectively.



Low quality scan - NLM filtered

Reduced noise and sharpened edges by Non-Local Means filter



Low quality scan - AI enhanced

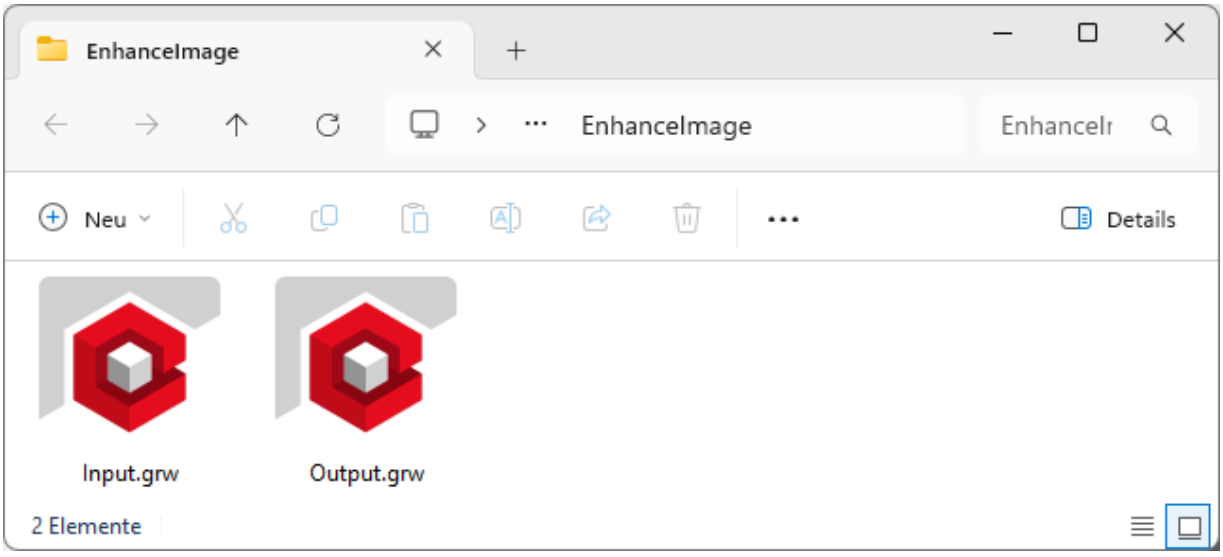
Scan after application of trained GeoDict-AI model

The NLM filter already reduces noise, but the contrast is worse compared to the scan with long exposure time. The AI model, however, produces a scan with a contrast comparable to the high quality scan while having less noise.

Once a model is trained with GeoDict-AI, short exposure times are sufficient to obtain high quality images using Enhance Image (AI) in GeoDict.

Result Folder

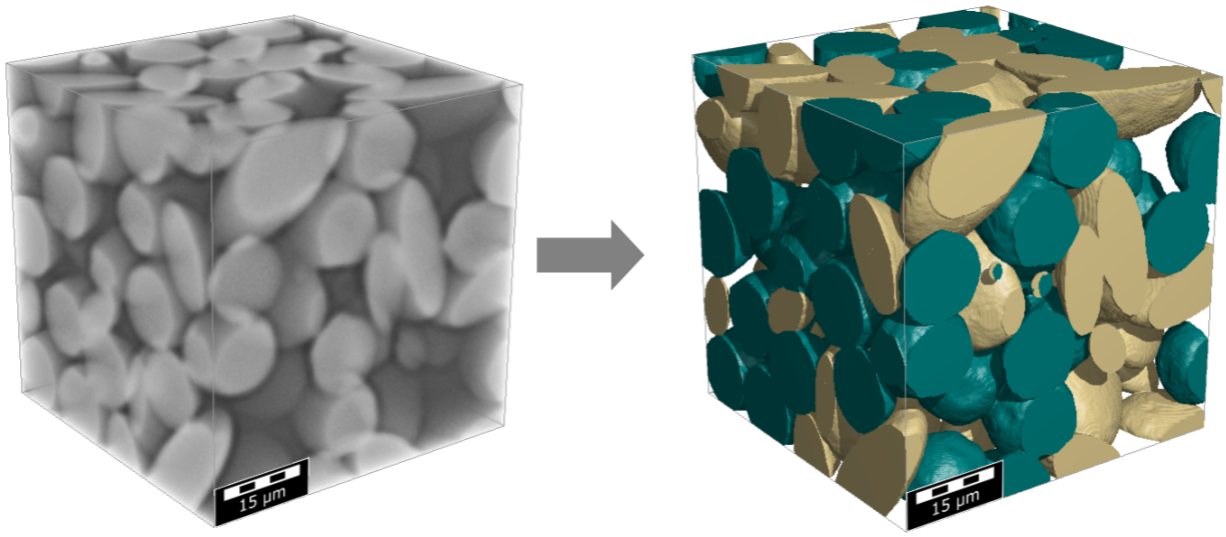
This folder contains the original image (**Input.grw**) and the resulting enhanced image (**Output.grw**).



1.8 Segment Grayscale Image

After producing training data 22 and training a neural network 39, the network can be applied to all images with statistical parameters fitting to the corresponding training data.

Use GeoDict-AI **Segment Grayscale Image**, to apply neural networks trained to segment 3D-scans (see also Taha and Hanbury, 2015 99).



The Segment Grayscale Image command:

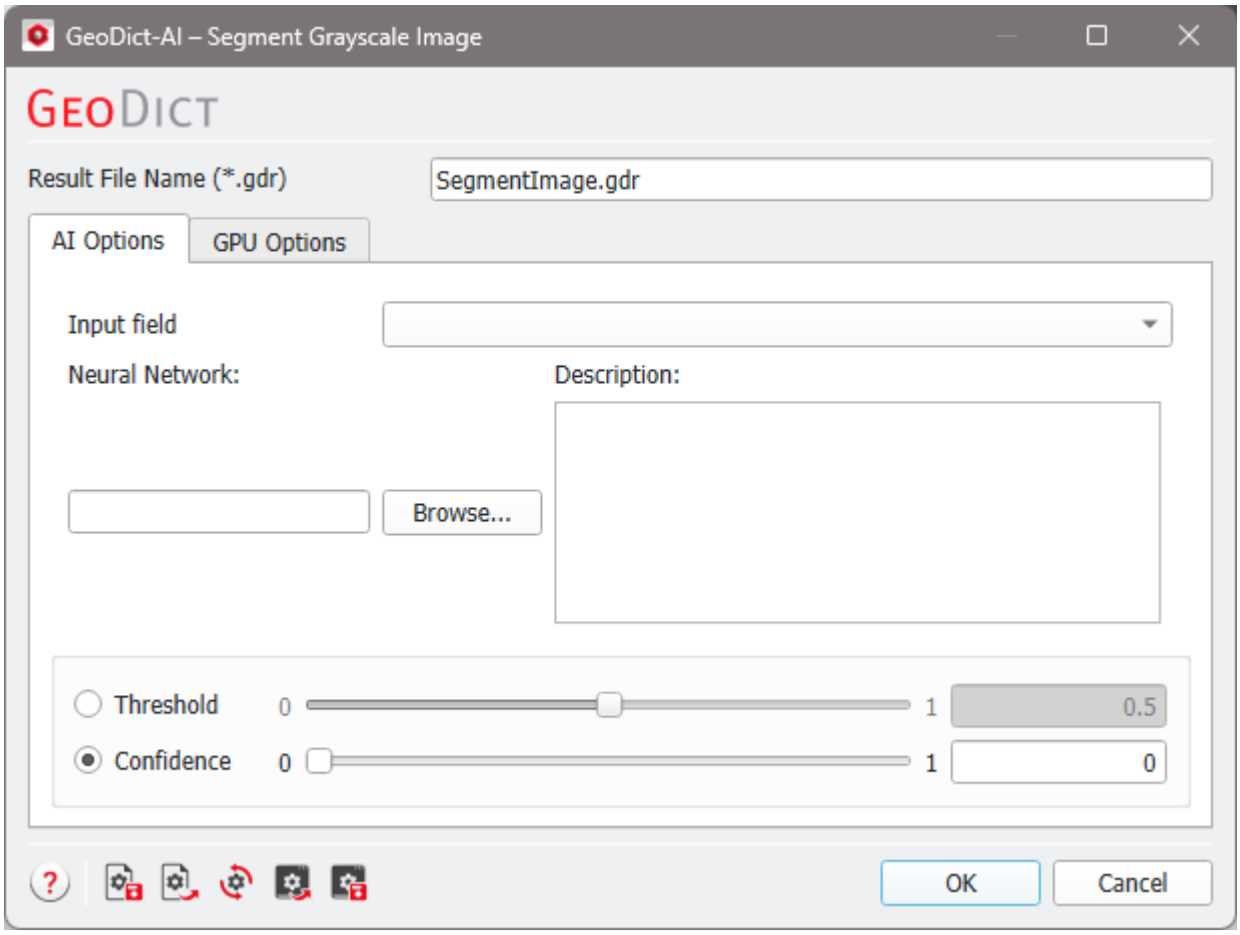
- Options 77
  Define the parameters for the image segmentation.
- Results 83
  Learn about the Segment Grayscale Image results.

1.8.1 Options

To start, load the image to analyze by selecting **File** → **Load Volume File ...** and browsing for the *.grw image to improve. Then, select **Segment Grayscale Image** from the pull-down menu in the GeoDict-AI section.

Click the **Options**' **Edit...** button.

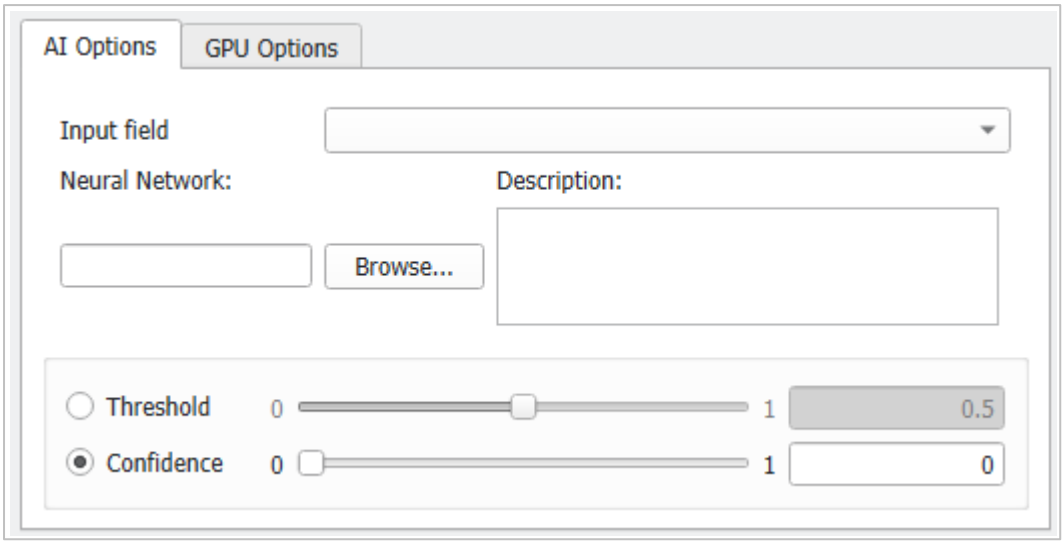
The **Segment Grayscale Image** dialog is organized in two tabs: **AI Options** and **GPU Options**.



Tabs of Segement Grayscale Image:

- AI Options 79
  Browse for the neural network to apply and select an image to segment.
- GPU Options 81
  Select a GPU and enter a batch size.

AI Options



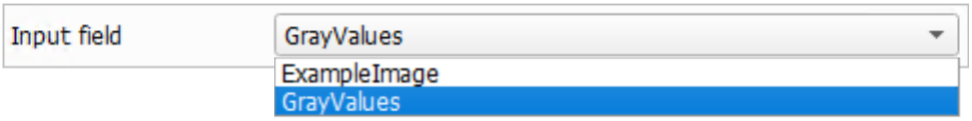
Initialization

Input Field

Select one of the currently loaded grayscale images as **Input field** to enhance from the pull-down menu.

Several grayscale images can be loaded at the same time by checking **Keep existing Volume Fields** in the Loading volume file dialog, when loading the image with **File - Load Volume File.** Using ImportGeo-Vol also different volume fields can be loaded.

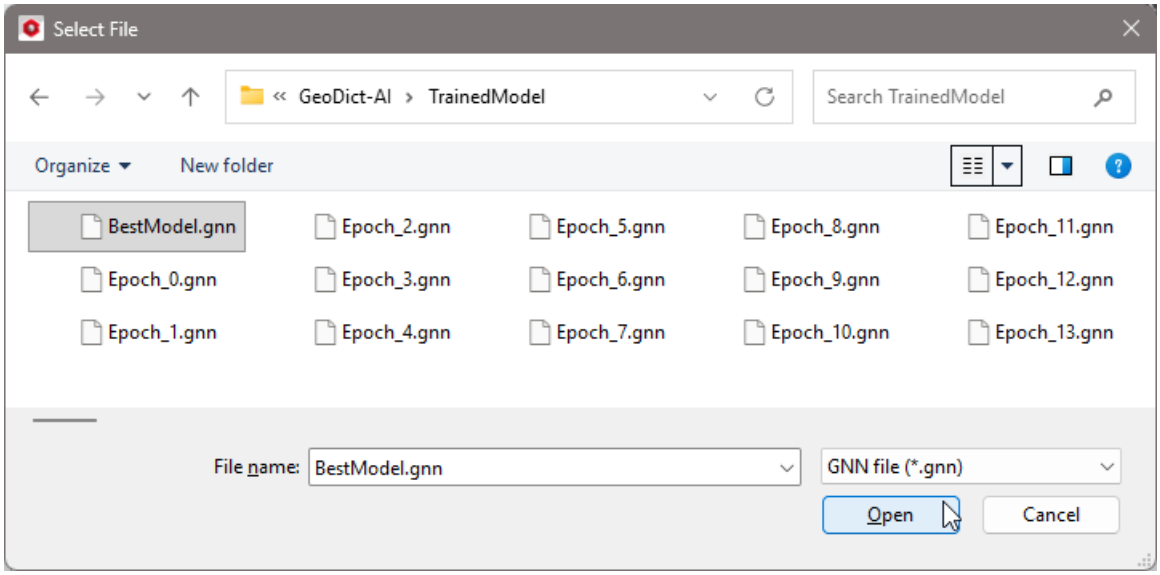
Then, select the **Input field** to segment from the pull-down menu.



Neural Network and Description

Browse for the neural network trained with **Train Neural Network** and most suitable for the image under consideration. For this, under **Neural Network** click **Browse**.

Select a neural network *.gnn file, for example the **BestModel.gnn** from a prior training.



In the **Description**, the current constraints for the application of the neural network are listed, as entered in the Train Neural Network 47 dialog, e.g., the diameter range of the fibers the neural network was trained for.

Transforming confidence field to structure

The neural network splits the structure into different material classes. For each resulting class a confidence field is computed. These fields contain values between 0 and 1. A value of 1 means, the neural network is perfectly sure, that this voxel should be assigned to the corresponding class, while a value of 0 means the voxel definitely does not belong to the corresponding class.

Each voxel then is labeled according to the field with the highest value. For example, consider a structure where three materials should be identified and the field values for one voxel are as follows:

- class 0 (cellulose fiber): 0.7
- class 1 (elliptical fiber): 0.25
- class 2 (binder): 0.05

In this case, the voxel is labeled as class 0 (cellulose fiber).

The parameters **Threshold** and **Confidence** are post-processing parameters applied after the neural network produced the confidence fields in the file nnOutput.npz. These parameters influence the labeling of voxels according to the probability values in the output fields (*.npz) from previous **Apply Neural Network** runs.

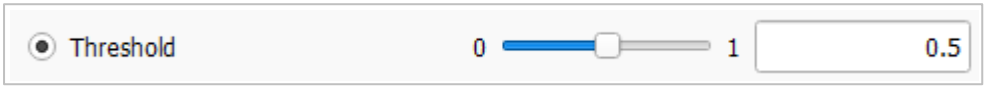
Use Threshold 81 for two-channel outputs, only, for example to prevent over-segmentation.

Use a higher Confidence[81] value if you want to find the voxels, where the network was uncertain.

Threshold

The **Threshold** allows proficient users to influence the decision to which material class a voxel belongs. The parameter is only available if the chosen network is trained to differentiate between exactly two material phases. For three or more material phases this parameter is grayed out.

If the identified features are over segmented, choosing a smaller threshold can lead to better results. For this, the confidence field (*.npz) can be loaded, without running the complete identification process again.



Find a visualization of the confidence field (*.npz) and threshold here[67].

For most applications, however, the default values can be kept because the default threshold of 0.5 usually produces good results for a well-trained network.

Important! This parameter only has an impact on two-channel results, i.e., distinguishing between only two material phases.

Confidence

Specify the minimum required confidence value for selecting a class for a voxel.

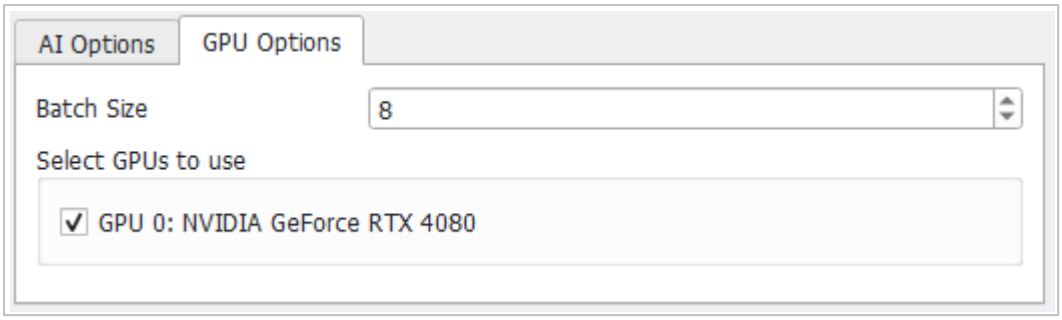
If a positive confidence value is specified and none of the fields has a higher probability value, the material is set to "undefined" in these uncertain locations, and material ID 255 is assigned. In the above[80] example, consider a confidence value of 0.8. In this case, the voxel would be labeled "undefined" instead of class 0.

The values of the confidence fields of all classes sum up to 1. Thus, it is impossible that for any voxel all confidence values are smaller than 1/n, where n is the number of classes. This is why setting the confidence value below 1/n has no impact. Therefore, the default value is set to 0, meaning all voxels are labeled as one of the classes as described above[80] and no uncertain areas will remain.

An example is shown here[67].

GPU Options

In the **GPU Options** panel define the parameters for the used graphic card(s).



Batch Size

The **Batch Size** is the number of samples, which the neural network uses simultaneously. For example, with the default batch size 4, during applying the network the error is computed, and the result updated for four windows at the same time, before moving on to the next four windows. In literature there is usually a warning about too large values for batch size due to the potential of overfitting, but in 3D image analysis here the batch size is limited by the available GPU memory. If the value is chosen too high, a warning dialog appears, when starting to apply the neural network. It is recommended to set the value as high as possible with the available GPU memory, so that it works without giving an error. This is often a value between 1 and 16 depending on the GPU hardware.

If multiple GPUs are selected, the batches are distributed to equal numbers on the different GPUs.

If the batch size was chosen too large for what is available on the graphics card, an error message appears suggesting to decrease the batch size.

Note! For CPU-based computation, the choice of batch size is not critical and can be left at the default.

Select GPUs to use

If a suitable graphics card is detected during installation of GeoDict, the GPU mode is used for applying the neural network, otherwise it is running in CPU mode.

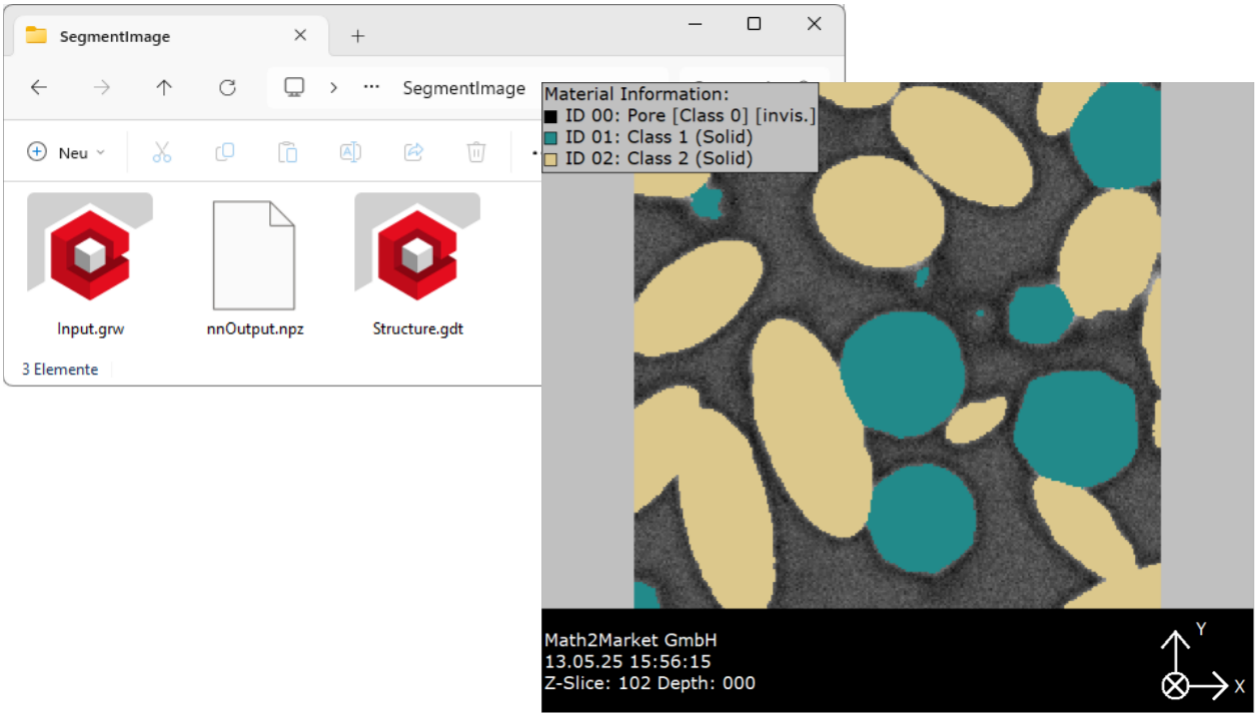
If multiple GPUs are available, select on which it should run. It can also run on multiple GPUs. If selecting more GPUs than licensed, an error message appears when clicking Run.

1.8.2 Results

After applying the neural network, a folder with the same name as the GeoDict result file is saved in the chosen project folder (**File** → **Choose Project Folder...** in the menu bar).

The result folder contains three files:

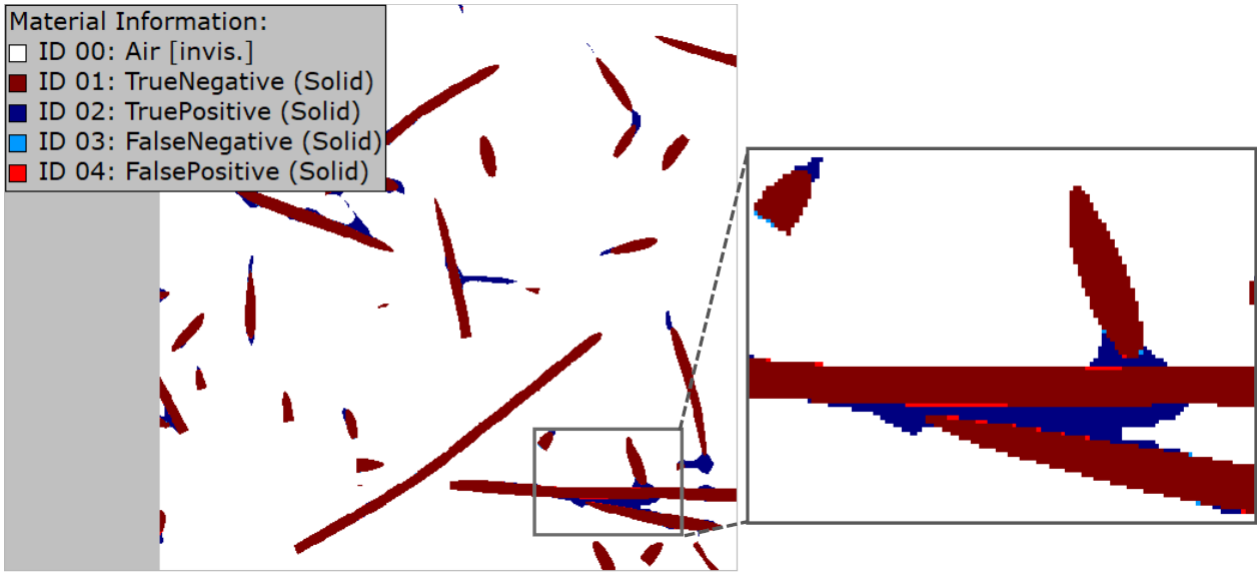
- **Structure.gdt**, the segmented structure
- **input.gdt**, the grayscale image that was segmented
- **nnOutput.npz**, the confidence field which is the neural network output.



1.9 Validate Performance

After training a neural network 39 **Validate** it's **Performance** before applying it to new structures. This command applies the given neural network to all input structures in a test structure folder (input.gdt) and compares it with their given ground truth (output.gdt).

Note! This option can only be used for neural networks that were trained to identify individual fibers or distinguish between material phases.



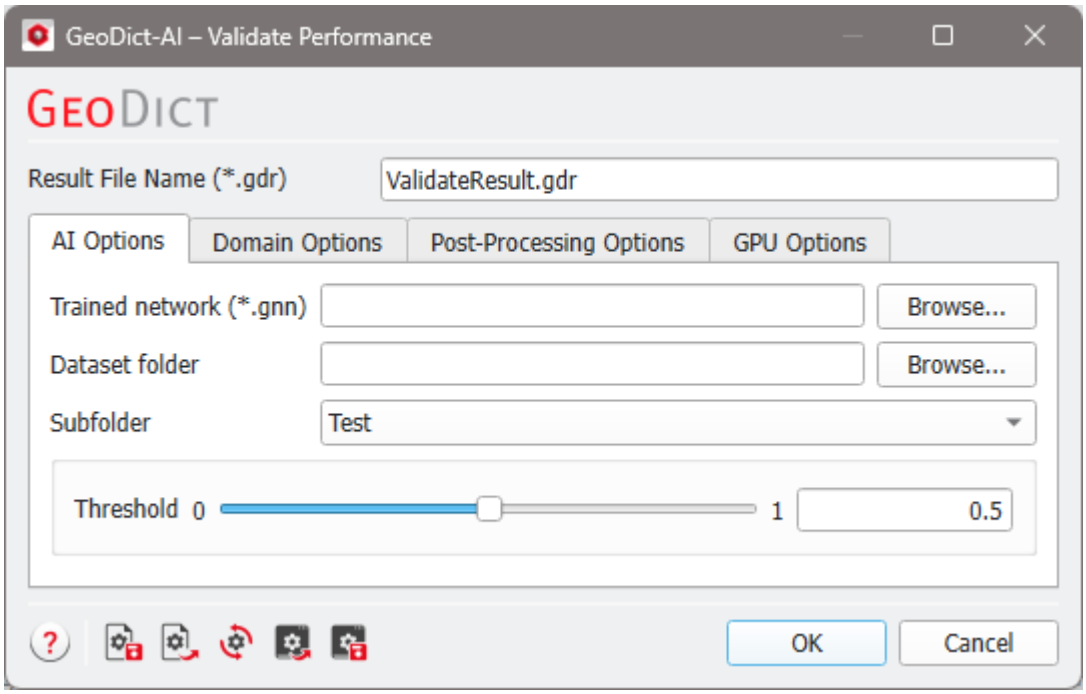
The Validate Performance command:

- Options 84
  Define the parameters for the network validation.
- Results 90
  Learn about the Validate Performance results.

1.9.1 Options

In the GeoDict-AI section select **Validate Performance** from the pull-down menu. The **GeoDict-AI Train Neural Network Options** dialog opens when clicking the **Options' Edit...** button in the GeoDict-AI section.

At the top of the **GeoDict-AI Validate** dialog, the name for the files containing the training results can be entered in the **Result File Name (*.gdr)** box. The default name can be kept, or a new name can be chosen fitting the current project.

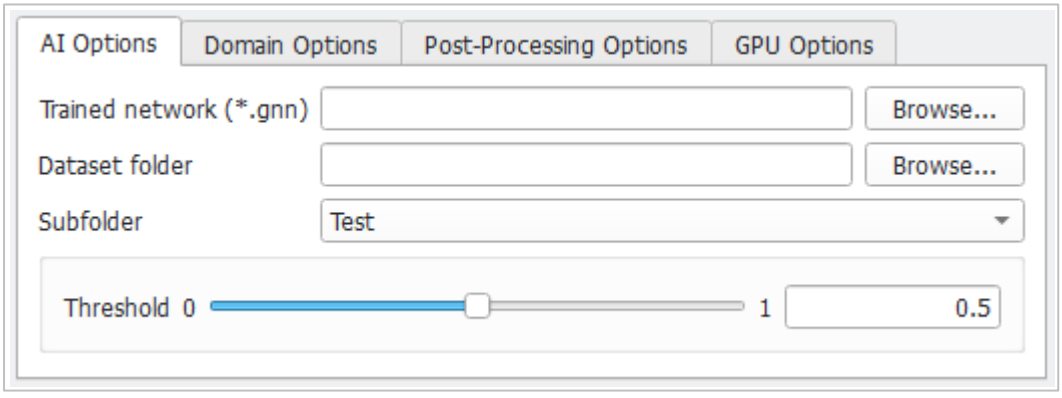


Tabs of Validate Performance:

- AI Options 85
  Browse for a neural network to validate and test data to analyze.
- Domain Options 86
  Decide if periodicity should be applied.
- Post-Processing Options 87
  Select the material that should be analyzed and if material phases or indiviual fibers should be identified.
- GPU Options 89
  Select a GPU and enter a batch size.

AI Options

Select which neural network should run on which data.



Trained Network

Browse for the **Trained Network (*.gnn)** to validate. Select the **BestModel.gnn** generated with Train Neural Network 55.

Dataset Folder

Choose the **Dataset Folder** created with **Create Training Data** containing the corresponding training data in the two folders **Train** and **Test**.

Subfolder

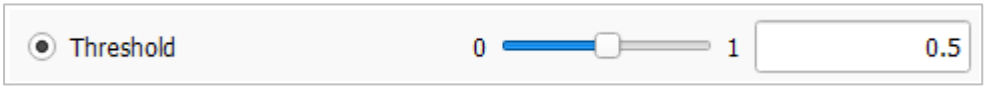
Select the **Subfolder** that should be used for validation:

- **Train** contains the data that was used for training. The neural network should perform well on this data, as it was explicitly trained for this.
- **Test** contains the data produced explicitly for the validation. This was not used in training and contains different data with similar statistical properties. If the network also performs well on this data, it can be applied on real scans with the required properties.

Threshold

The **Threshold** allows proficient users to influence the decision to which material class a voxel belongs. The parameter is only available if the chosen network is trained to differentiate between exactly two material phases or to identify fibers. For three or more material phases this parameter cannot be used.

If the identified features are over segmented, choosing a smaller threshold can lead to better results. For this, the confidence field (*.npz) can be loaded, without running the complete identification process again.



Find a visualization of the confidence field (*.npz) and threshold here 67.

For most applications, however, the default values can be kept because the default threshold of 0.5 usually produces good results for a well-trained network.

Important! This parameter only has an impact on two-channel results, i.e., distinguishing between only two material phases.

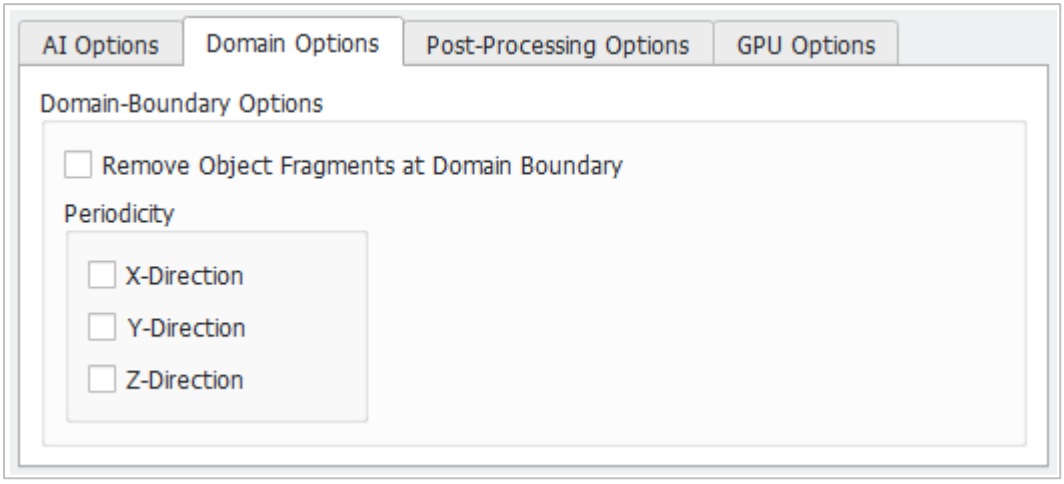
Domain Options

Select to **Remove Object Fragments at Domain Boundary** if boundary objects could falsify the statistics. This setting only has an effect for the post-processing option Fibers

(Centerlines) during the Identify Fibers run. For material phases this setting is ignored and objects are not removed.

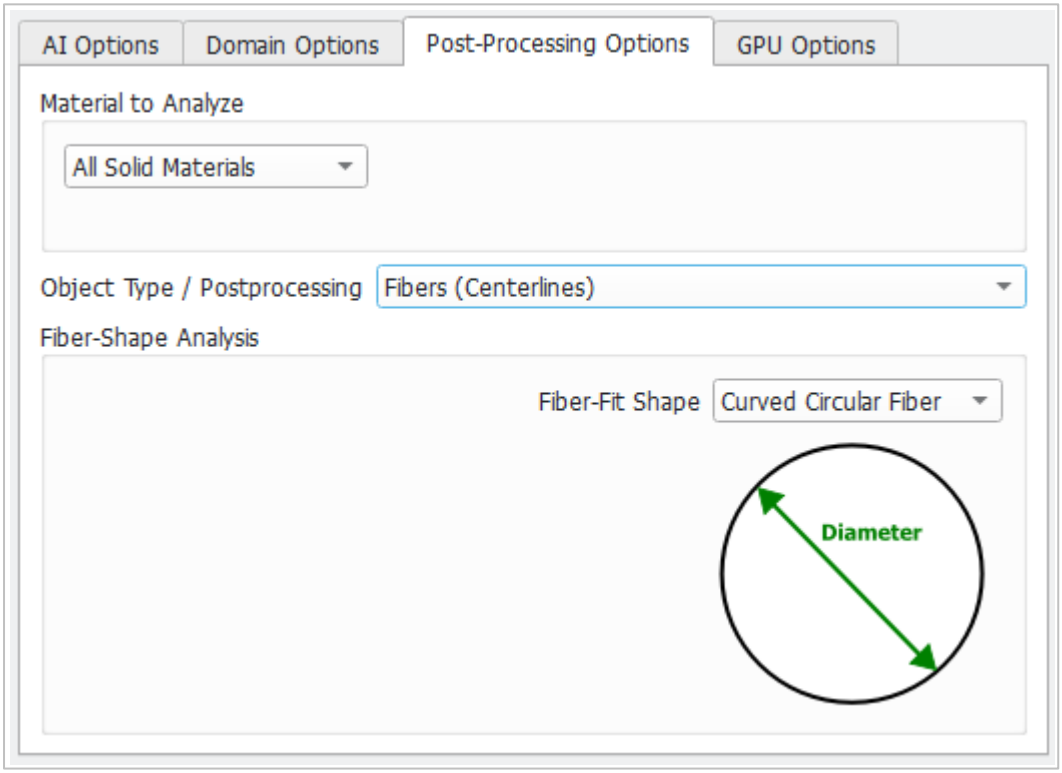
The underlying domain can bet set as periodic for the material or fiber identification by checking one or several of the boxes in the **Domain-Boundary Options** panel.

However, periodicity is unusual for most 3D scans and this setting is rarely changed.



Post-Processing Options

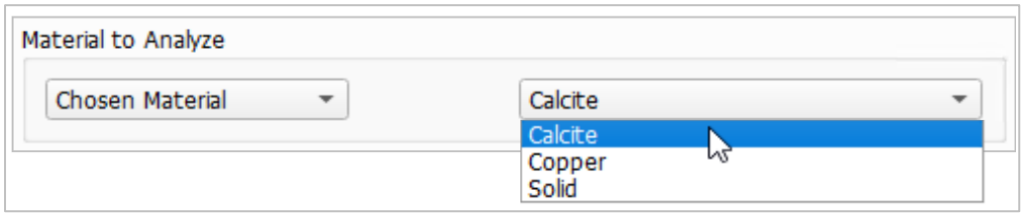
The parameters under the **Post-Processing Options** define how the neural network will be applied.



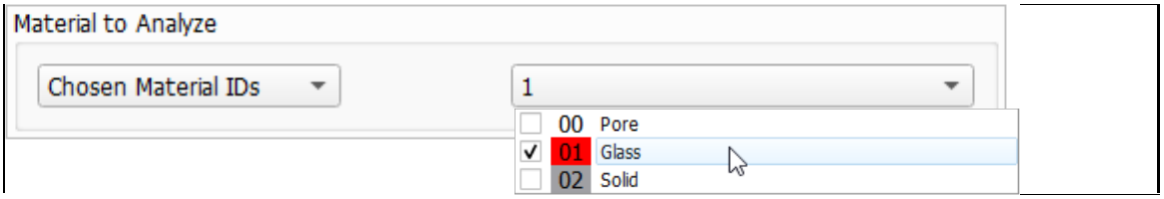
Material to Analyze

The **Material to Analyze** panel offers the choice of whether GeoDict-AI should be applied to all solids in the structure or only on a subset, defined either by choosing material IDs or materials.

If **Chosen Material** is selected, select the material to analyze from the pull-down menu on the right. All other materials will be considered as background by the neural network.



If **Chosen Material IDs** is selected, choose the material IDs to analyze from the pull-down menu on the right. All other material IDs will be considered as background by the neural network.



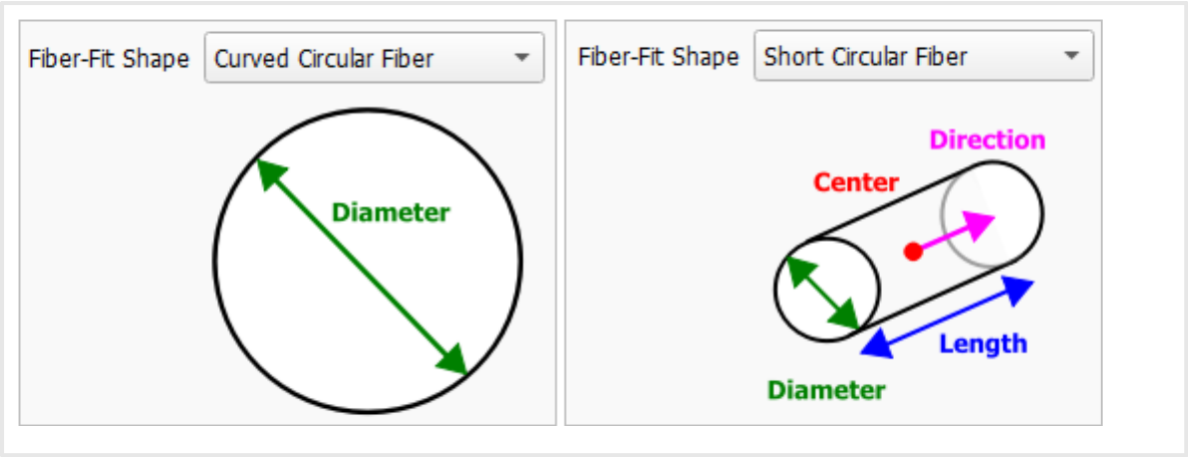
Object Type / Postprocessing

Define the **Object Type / Postprocessing** that was used for creating the training data. Select **Fibers (Centerlines)** if the network was trained to identify the individual fibers. If instead different materials as for example fibers and binder are distinguished, select **Material Phases**.

Fibers (Centerlines) can only be applied, if a valid FiberFind license is available.

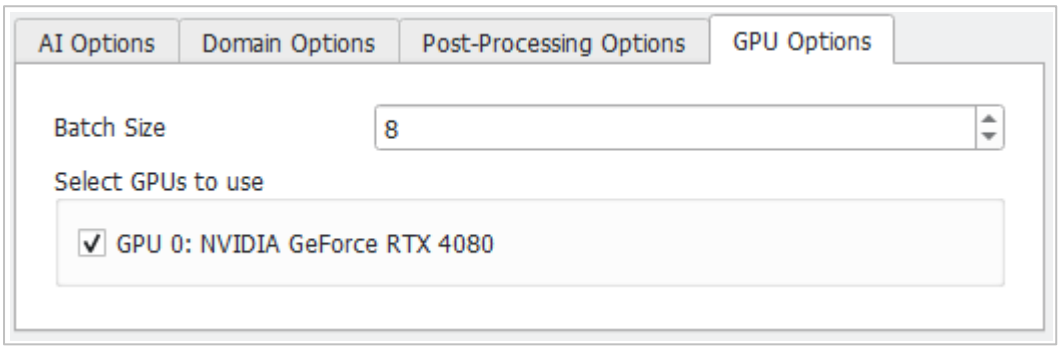
Fiber-Fit Shape

If **Fibers (Centerlines)** is selected, decide on the **Fiber-Fit Shape** from the pull-down menu, depending on the training data. In the bottom right corner, **Curved Circular Fiber** and **Short Circular Fiber** are available.



GPU Options

In the **GPU Options** panel define the parameters for the used graphic card(s).



Batch Size

The **Batch Size** is the number of samples, which the neural network uses simultaneously. For example, with the default batch size 4, during applying the network the error is computed, and the result updated for four windows at the same time, before moving on to the next four windows. In literature there is usually a warning about too large values for batch size due to the potential of overfitting, but in 3D image analysis here the batch size is limited by the available GPU memory. If the value is chosen too high, a warning dialog appears, when starting to apply the neural network. It is recommended to set the value as high as possible with the available GPU memory, so that it works without giving an error. This is often a value between 1 and 16 depending on the GPU hardware.

If multiple GPUs are selected, the batches are distributed to equal numbers on the different GPUs.

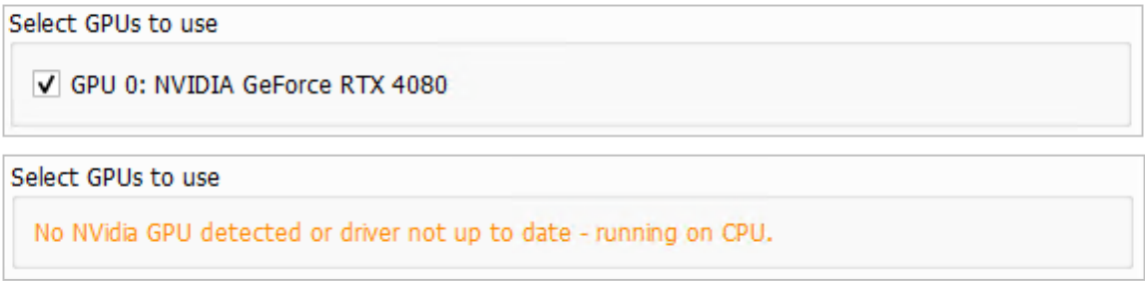
If the batch size was chosen too large for what is available on the graphics card, an error message appears suggesting to decrease the batch size.

Note! For CPU-based computation, the choice of batch size is not critical and

can be left at the default.

Select GPUs to use

If a suitable graphics card is detected during installation of GeoDict, the GPU mode is used for applying the neural network, otherwise it is running in CPU mode.



If multiple GPUs are available, select on which it should run. It can also run on multiple GPUs. If selecting more GPUs than licensed, an error message appears when clicking Run.

1.9.2 Results

Running **Validate Performance** produces a folder with the same name as the GeoDict result file. Both are saved in the chosen project folder (**File** → **Choose Project Folder...** in the menu bar).

Report

The GeoDict **Result Viewer** opens for the result file and the **Results – Report** subtab shows the **Confusion Matrix & IoU Score** for each analyzed structure. Learn more about the confusion matrix here 52.

Structure 15

Confusion Matrix & IoU Score

	Predicted Class 1	Predicted Class 2	IoU Score
Actual Class 1	1.00	0.00	0.99
Actual Class 2	0.04	0.96	0.93

If **Fibers** was chosen for **Object Type**, the report additionally shows statistics regarding the incorrectly identified fibers for each structure used in the validation. Below find information about the **Split objects** - fibers that were incorrectly split in two or more fibers.

Incorrectly identified objects (error > 1%)

Split objects

Component ID	Component volume	Biggest matching component volume fraction	Matching components (id/volume)
738	3145	0.96	(842/3025), (23/88), (33/22), (345/10)
314	22719	0.99	(818/22424), (794/293), (809/1), (883/1)

Next find a table for the **Merged objects** - fibers incorrectly merged into one.

Merged objects

Component ID	Component volume	Biggest matching component volume fraction	Matching components (id/volume)
287	8949	0.60	(386/5382), (623/3567)
588	10633	0.81	(772/8617), (657/2015), (10/1)
69	9740	0.84	(140/8145), (676/1591), (693/4)
345	16753	0.94	(200/15701), (718/1044), (15/8)
73	22172	0.94	(143/20791), (439/1380), (384/1)
402	17957	0.94	(312/16867), (184/1087), (150/2), (91/1)
721	2866	0.94	(531/2695), (431/171)
319	31928	0.96	(743/30641), (104/1267), (298/18), (511/1), (688/1)
18	5885	0.96	(454/5653), (196/219), (173/12), (458/1)

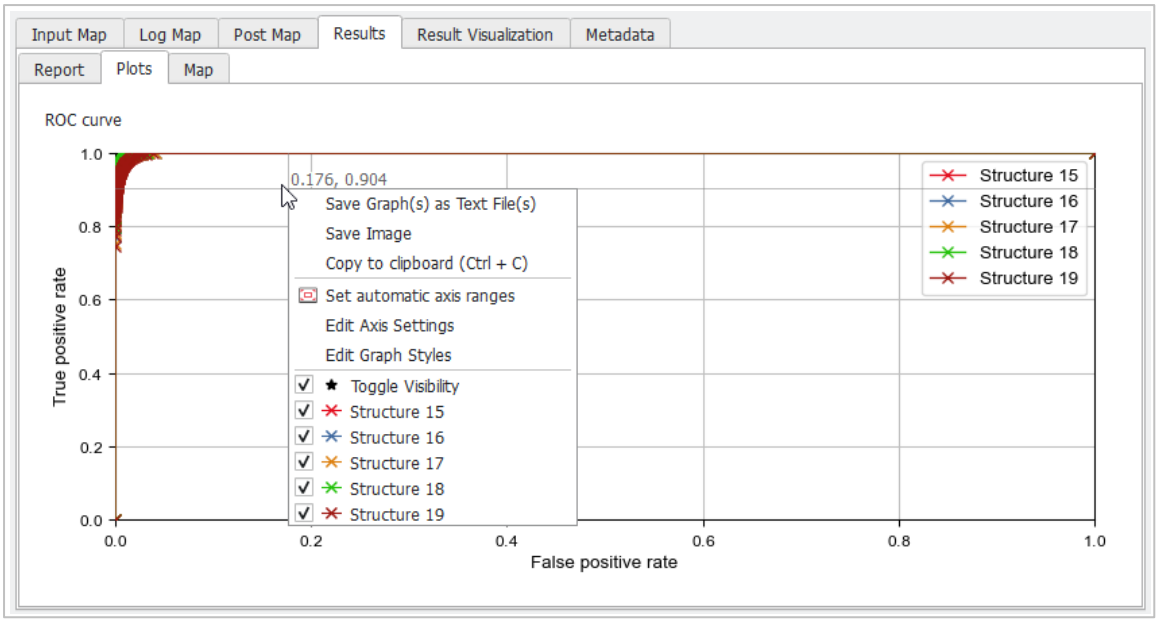
Also the **Lost Objects** are listed. These are fibers, where parts of them were not detected.

Lost objects

Component ID	Component volume
47	512
67	1049
69	185
72	749
126	1029
161	732

Plots

For two-channel cases, the **Plots** subtab shows the **Receiver Operating Characteristic** (**ROC**) curve for each selected structure.



Right-clicking in the plot opens a dialog offering many possibilities to change the plot settings or to save the data. Refer to the Result Viewer chapter to learn more about these options.

When applying a neural network, a confidence field is generated. This field then is segmented with the **Threshold** given in the Apply Neural Network Options 59 dialog.

Usually, not all voxels are identified correctly. The proportion of voxels correctly assigned to the target material, then is called the **True positive rate**. The proportion of voxels wrongly assigned to the target material is called the **False positive rate**.

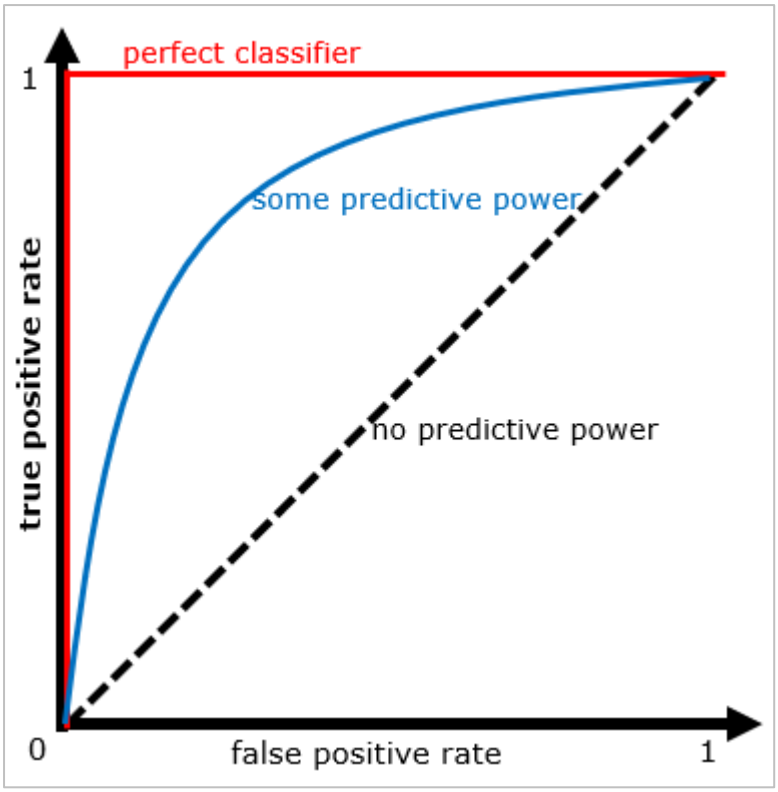
For a threshold of 0 all solid voxels are assigned to the target material. Thus, the **True positive rate** and the **False positive rate** are both 1. For a threshold of 1 no voxels are assigned to the target material and thus, both rates are 0.

The **ROC curve** shows how the relation of **True positive rate** and **False positive rate** changes if varying the threshold from 0.0 to 1.0.

If the curves look as above, the neural network performs very well on the structures.

The optimum threshold is the upper left area, as the true positive rate is high, but the false positive rate is still low.

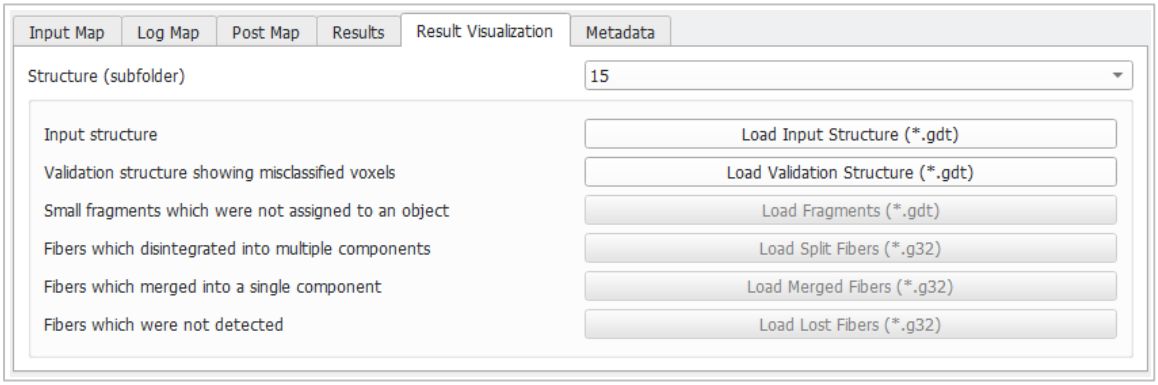
The values should lie above of the main diagonal, as otherwise the performance is not better than simply guessing the material IDs. If for all thresholds in (0,1) the true positive rate is 1 and the false positive rate is 0, the network is a perfect classifier, i.e., the result is always perfectly identified materials for all possible thresholds. This means, that the confidence field only contains values near 0 and values near 1, as the neural network is very confident. Usually, obtaining a perfect classifier is not realistic. But it can be close, as in the example above.



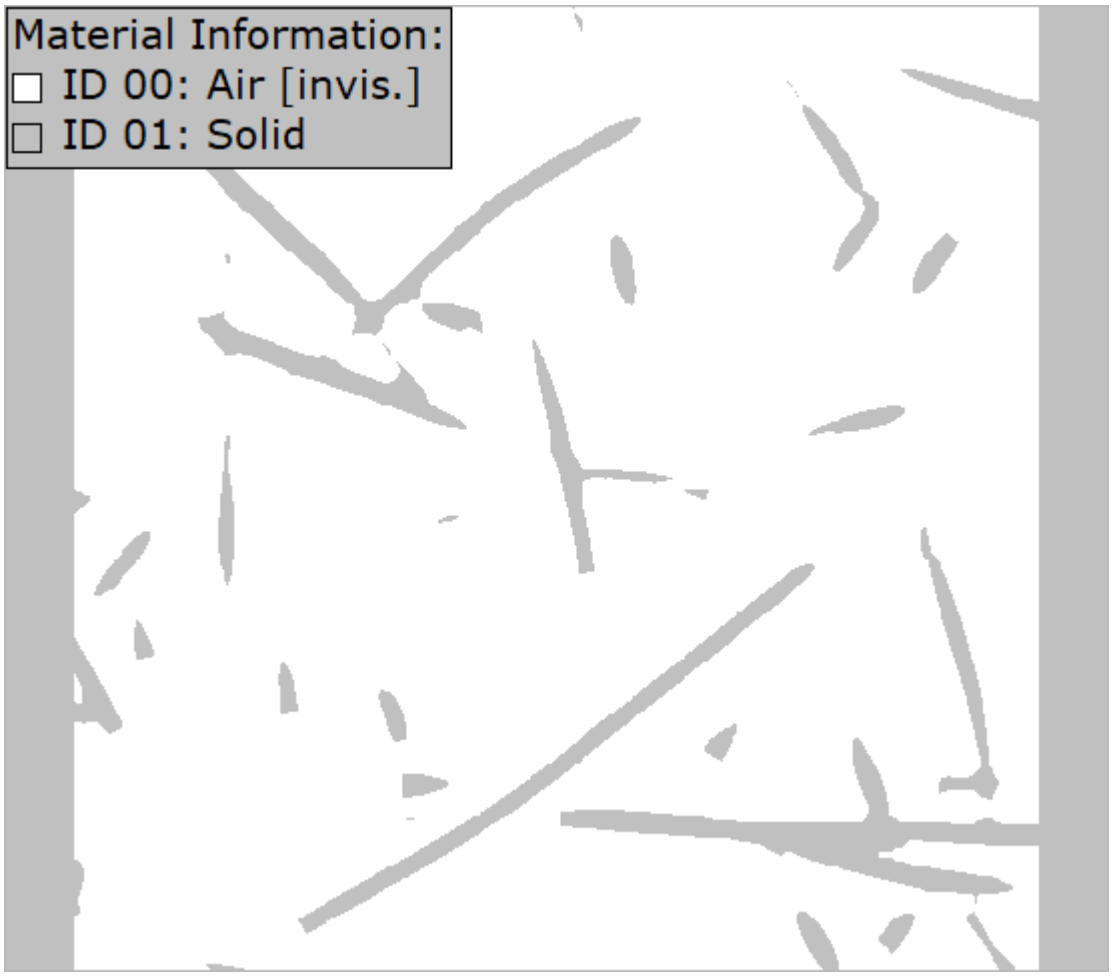
Result Visualization

The **Result Visualization** tab provides several 3D visualization options for the results. From the pull-down menu **Structure (subfolder)** select the subfolder from which the results should be visualized.

If **Material Phases** was selected for **Object Type / Postprocessing** in the panel below only the **Input structure** and the **Validation structure showing misclassified voxels** are available.



Clicking **Load Input Structure (*.gdt)** loads the file **Input.gdt** only containing two material IDs: ID 00 for the background voxels, usually the pore space, and ID 01 for the solid materials.



Click **Load Validation Structure (*.gdt)** to load the file **Validation.gdt**.

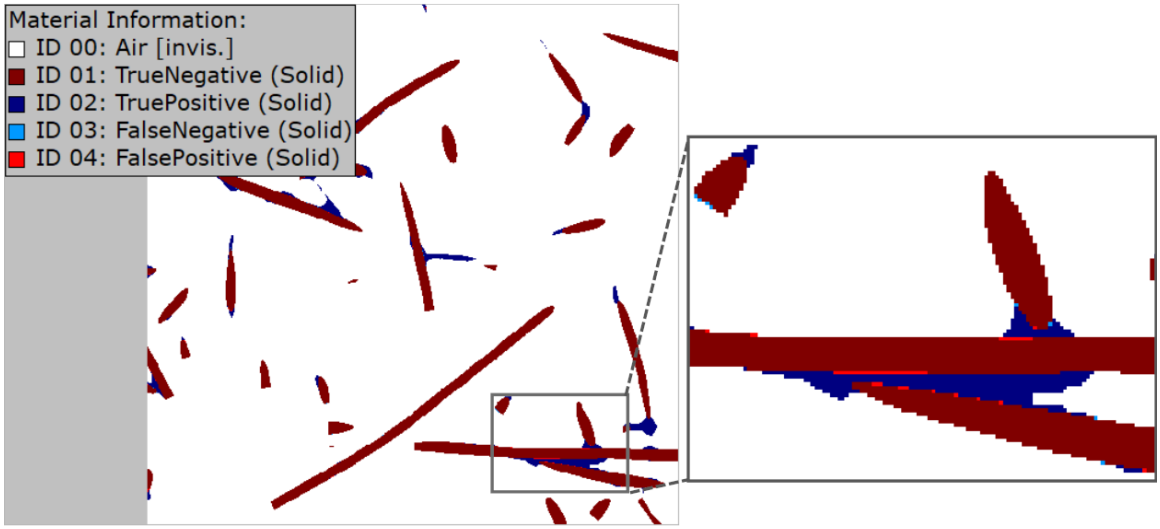
For two-channel cases, five material IDs are present in this structure and the colors assigned to the material IDs are changed automatically after loading the validation structure:

- **ID 00 Pore** (invisible): The pore space that is not considered for the identification of target material, in the example it is assigned to air, according to the material IDs in the input structure.
- **ID 01 TrueNegative** (dark red): the solid voxels that the neural network correctly identified as no target material.
- **ID 02 TruePositive** (dark blue): the solid voxels that are correctly identified as target material.
- **ID 03 FalseNegative** (light blue): the solid voxels that are target material, but are not identified as it.
- **ID 04 FalsePositive** (light red): the solid voxels that are wrongly identified as target material.

These terms come from the confusion matrix commonly used to describe the quality of a neural network.

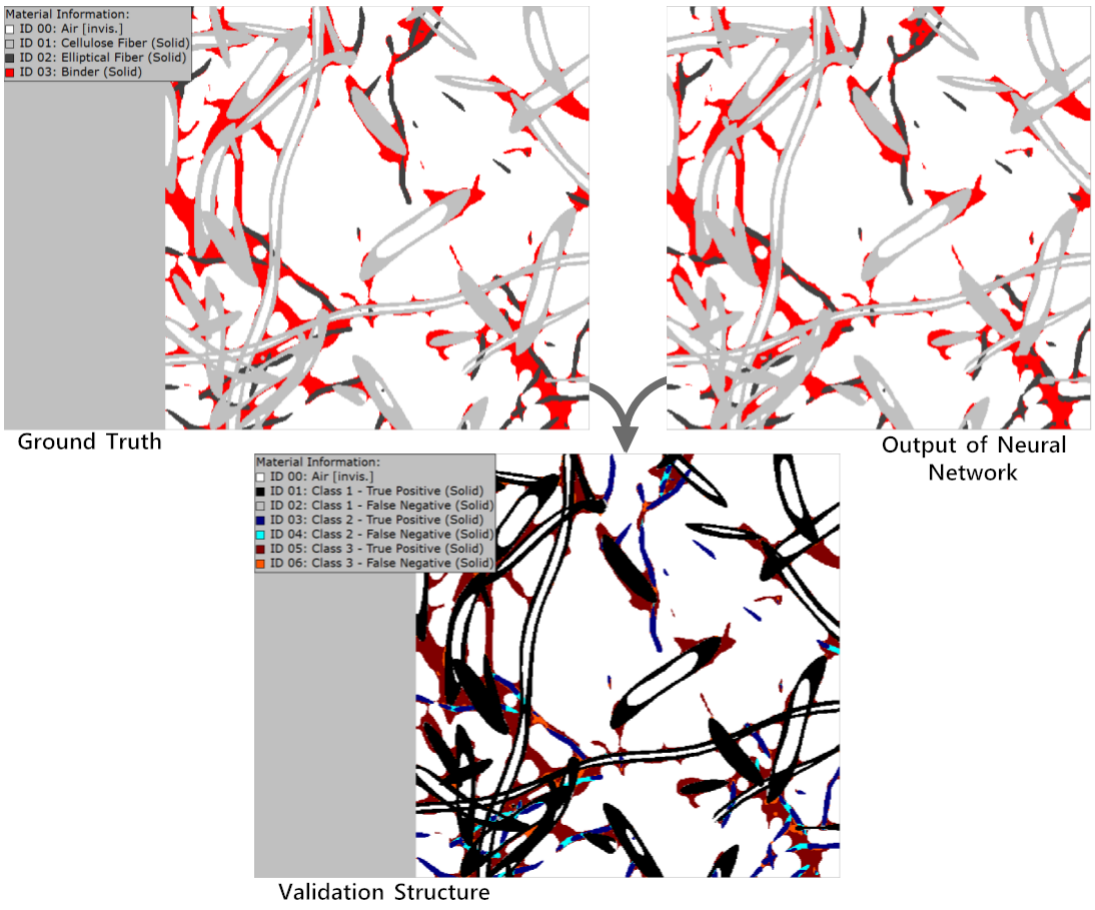
		Actual Values: Positive	Actual Values: Negative
Predicted Values	Positive	TP	FP
	Negative	FN	TN

In the image below, the material phases binder and fibers were distinguished. The voxels correctly identified as fiber (**TrueNegative**) are visualized in dark red and the correct identified binder in dark blue (**TruePositive**). The light blue voxels show, where binder was identified as fiber (**FalseNegative**) and the fibrous voxels identified as binder are marked in light red (**FalsePositive**). In this example, the neural network identified the binder really well. Only a few voxels at the boundary between fiber and binder were identified wrongly.



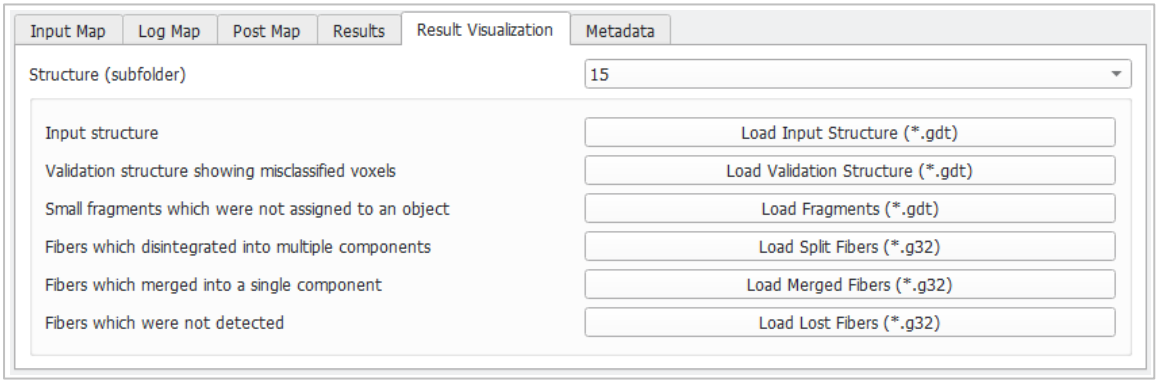
For multi-channel cases, each channel gets two material IDs:

- **Class # - True Positive**: correct identified as class #.
- **Class # - False Negative**: wrongly not identified as class #.

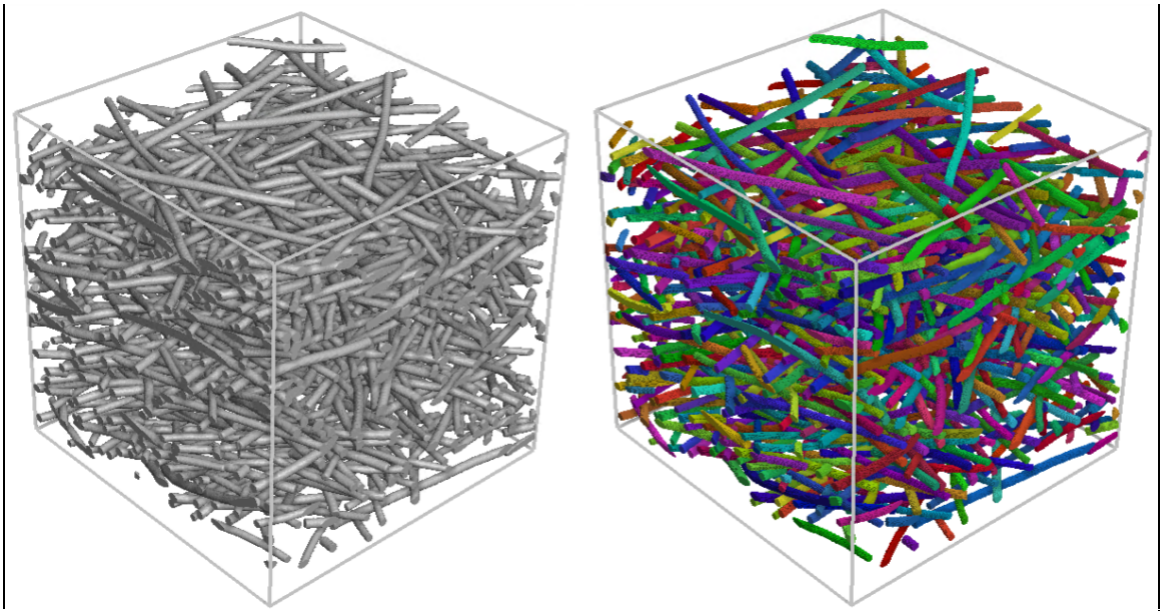


If **Fibers** was selected for **Object Type**, additional files are available:

- Small fragments which were not assigned to an object
- Fibers which disintegrated into multiple components
- Fibers which merged into a single component.
- Fibers which were not detected



In the following figure, the input fiber structure and the result from FiberFind are shown. The FiberFind result files and result folders can also be found in the **Validate Performance** result folder 97.

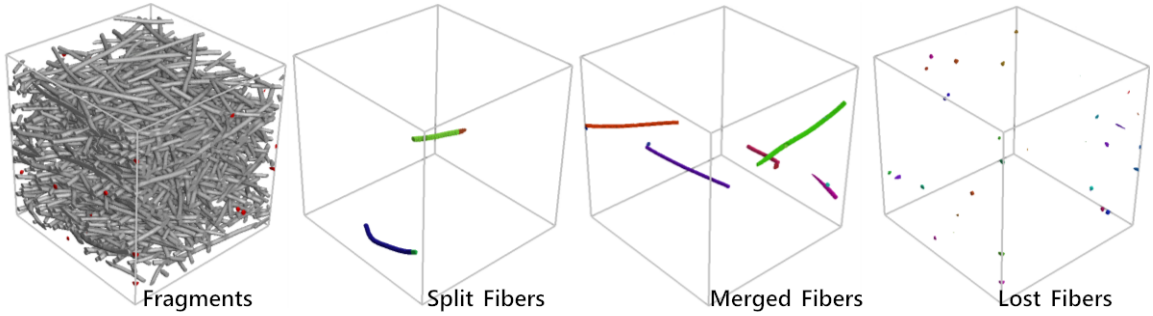


To check, if there are fragments of solid material not identified as fibers, click **Load Fragments (*.gdt)** (see the first image below).

Click **Load Split Fibers (*.g32)** to observe if fibers were split into multiple components (see the second image below).

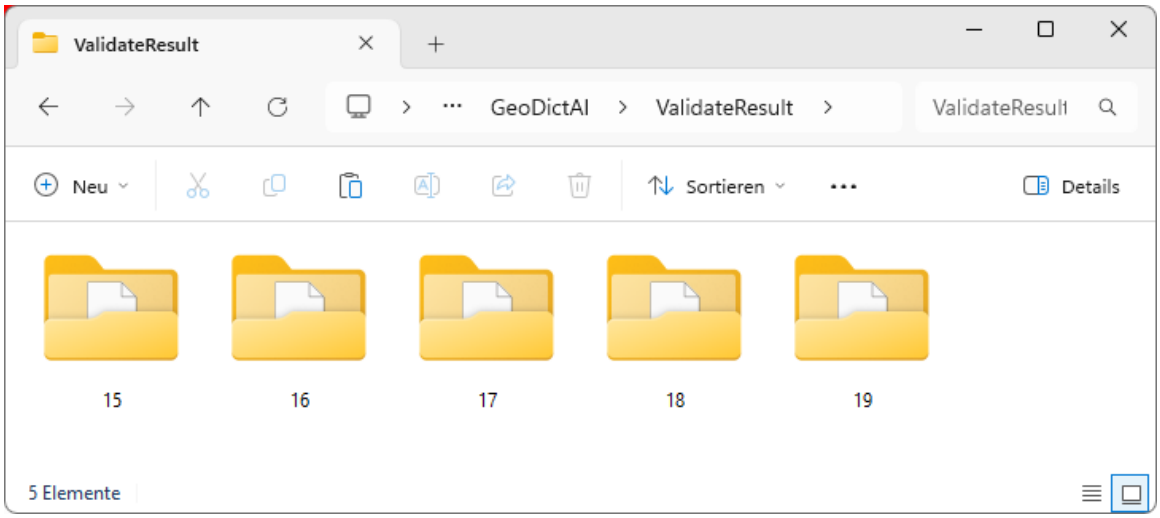
Also, different fibers can have been merged into one fiber. Click **Load Merged Fibers (*.g32)** to check on it (see the third image below).

There can be fiber parts not detected as fibers. Click **Load Lost Fibers (*.g32)** to visualize them (see the last image below).



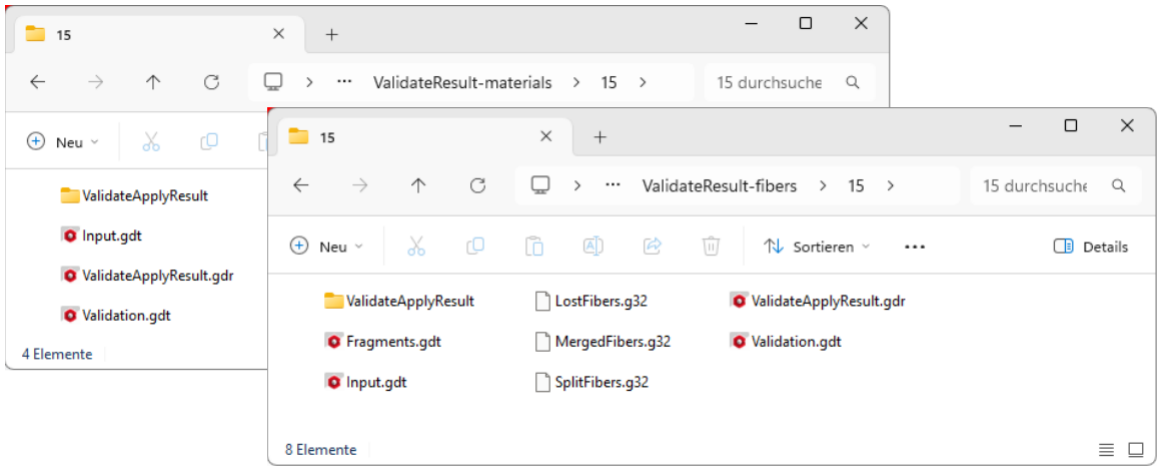
Result Folder

The result folder contains the same number of folders as the folder selected for **Subfolder** as described above, each folder corresponding to a different training data set.



These folders then contain the following data:

- The file **Input.gdt** is copied from the training data 35.
- The file **ValidateApplyResult.gdr** and the corresponding folder ValidateApplyResult are generated by applying the neural network to Input.gdt.
- The file **Validation.gdt** contains the validation result visualization 93.
- The file **Fragments.gdt** is only available for the Object Type 88 option Fibers. It contains the fragments not assigned to a fiber.
- The file **LostFibers.g32** is only available for the Object Type 88 option Fibers. It contains the fibers not detected.
- The file **MergedFibers.g32** is only available for the Object Type 88 option Fibers. It contains the wrongly merged fibers.
- The file **SplitFibers.g32** is only available for the Object Type 88 option Fibers. It contains the wrongly split fibers.



1.10 References

V. Chandra, A. I. Al-Naimi, *Petroleum Engineering Research Center*, King Abdullah University of Science and Technology

D.P. Kingma, J. Ba, *Adam: A method for stochastic optimization*, arXiv preprint arXiv:1412.6980 (2016), https://doi.org/10.48550/arXiv.1412.6980

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala, *PyTorch: An Imperative Style, High-Performance Deep Learning Library*, arXiv:1912.01703 [cs.LG] (2019), https://doi.org/10.48550/arXiv.1912.01703

O. Ronneberger, P. Fischer, T. Brox, *U-Net: Convolutional Networks for Biomedical Image Segmentation*, arXiv:1505.04597 (2015), https://doi.org/10.48550/arXiv.1505.04597

S. Ruder, *An overview of gradient descent optimization algorithms*, arXiv:1609.04747 (2016), https://doi.org/10.48550/arXiv.1609.04747

I.M. Sobol, *On the Distribution of points in a cube and the approximate evaluation of integrals*, USSR Computational Mathematics and Mathematical Physics (1967), https://doi.org/10.1016/0041-5553(67)90144-9

A. A. Taha, A. Hanbury, *Metrics for evaluating 3D medical image segmentation: analysis, selection, and tool*, BMC Medical Imaging (2015), https://doi.org/10.1186/s12880-015-0068-x

Citation: Math2Market GmbH, 2026: GeoDict 2026 User Guide, GeoDict-AI, Math2Market GmbH, Germany, doi.org/10.30423/userguide.geodict

MATH 2 MARKET

Math2Market GmbH
Richard-Wagner-Str. 1, 67655 Kaiserslautern / Germany
www.math2market.com