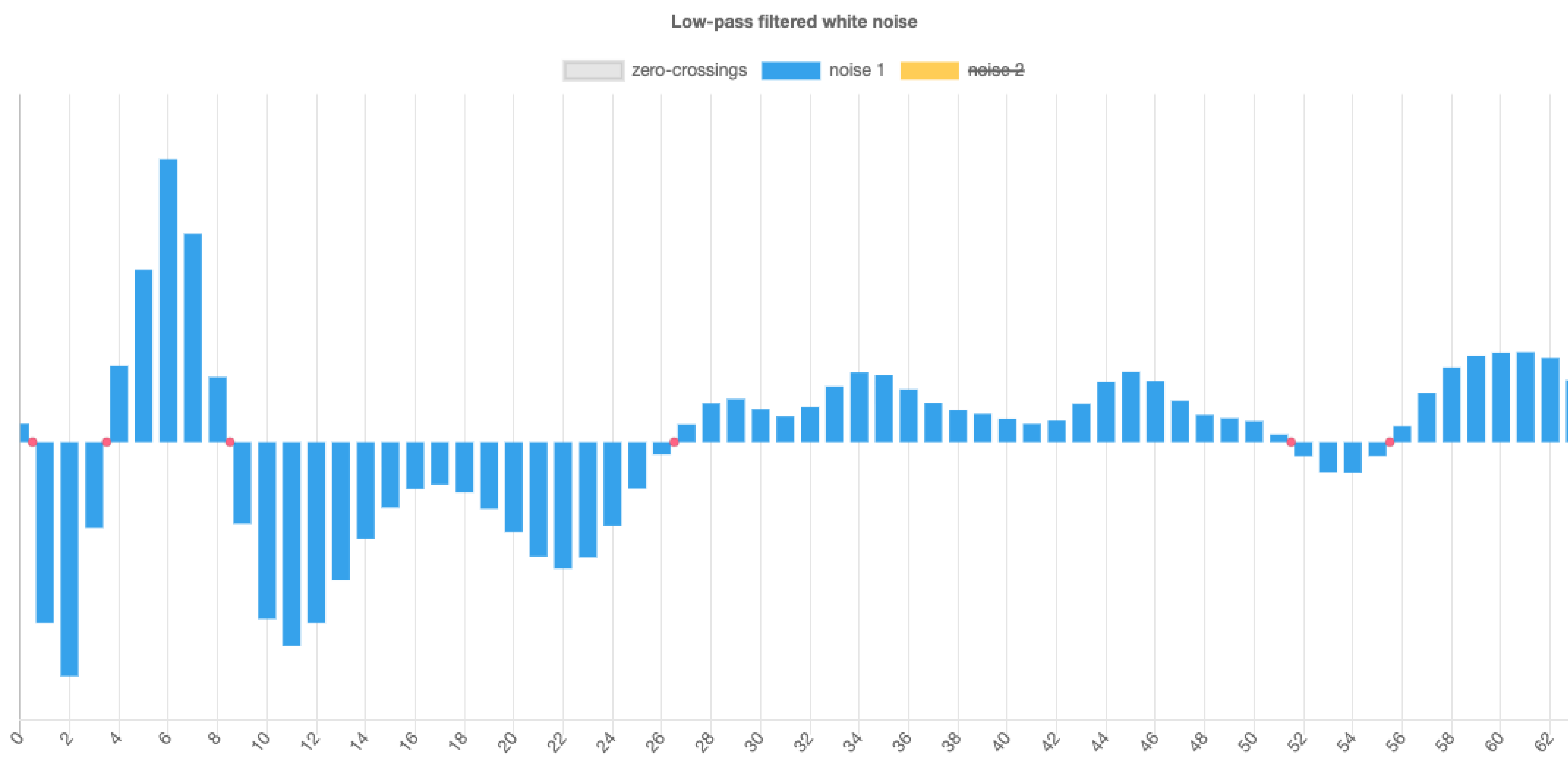
What can('t) we do with low-pass filtered white noise?

Here an illustration of the vessel generation process. Following Edwin Abbott's Flatland, in order to see what is going on we can start with a simple 1-D case. We create an array with 64 random numbers between -0.5 and +0.5. The bars in the graph below represent the values at 64 line element positions. For positive values the bar goes up, for negative values the bar goes down. Each time the random number is close to zero the bar is shortest. The red dots represent positions on the line where the sign of the bars change, each time a bar is followed by another bar with the opposite sign. These points indicate the crossings of the zero-line of our random list of numbers.



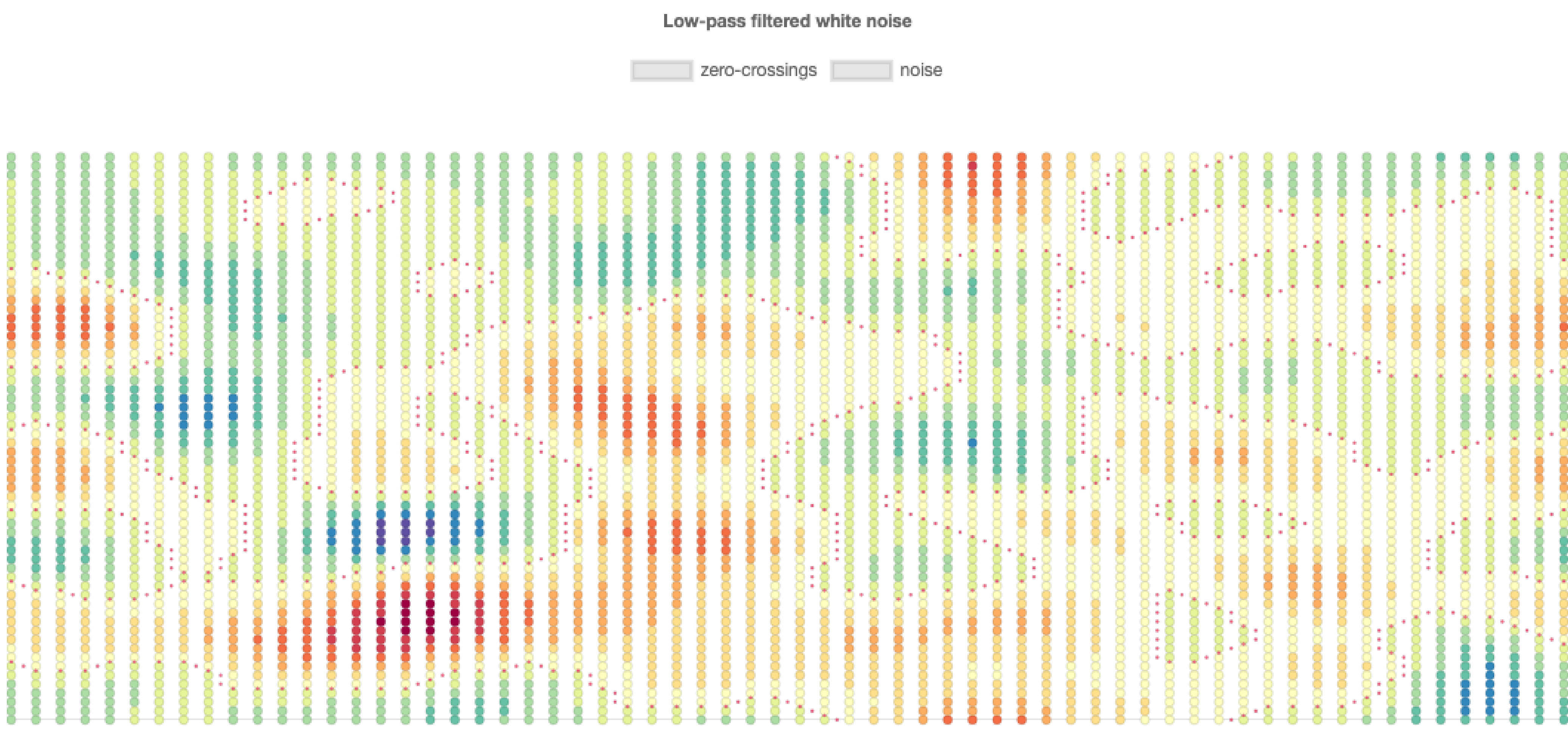
Low-pass filter (iteration: 17)

Each time we click on the "Low-pass filter" button the initially random list of numbers is smoothed by averaging three consecutive values. As this step is repeated only low frequency variations in the data are retained. High frequency components are removed. In order to show this low-pass effect each click applies the filter initially once later 7 times. As the number of filtering steps increases the values become smaller (rescaling is used for display) and neighboring values become more similar to each other. We see larger structures dominate with fewer and fewer zero-crossings.

There is a second (hidden) random array of numbers. If you enabled the display of "noise 2" in the figure you can see that similar observations apply to this independently smoothed array. It also becomes smoother and positive-sign (up) and negative (down) regions are established.

Summary: In 1-D a low-pass filtered white noise array has zero-crossings that are points (have dimension 0). As more smoothing steps are applied fewer points are generated.

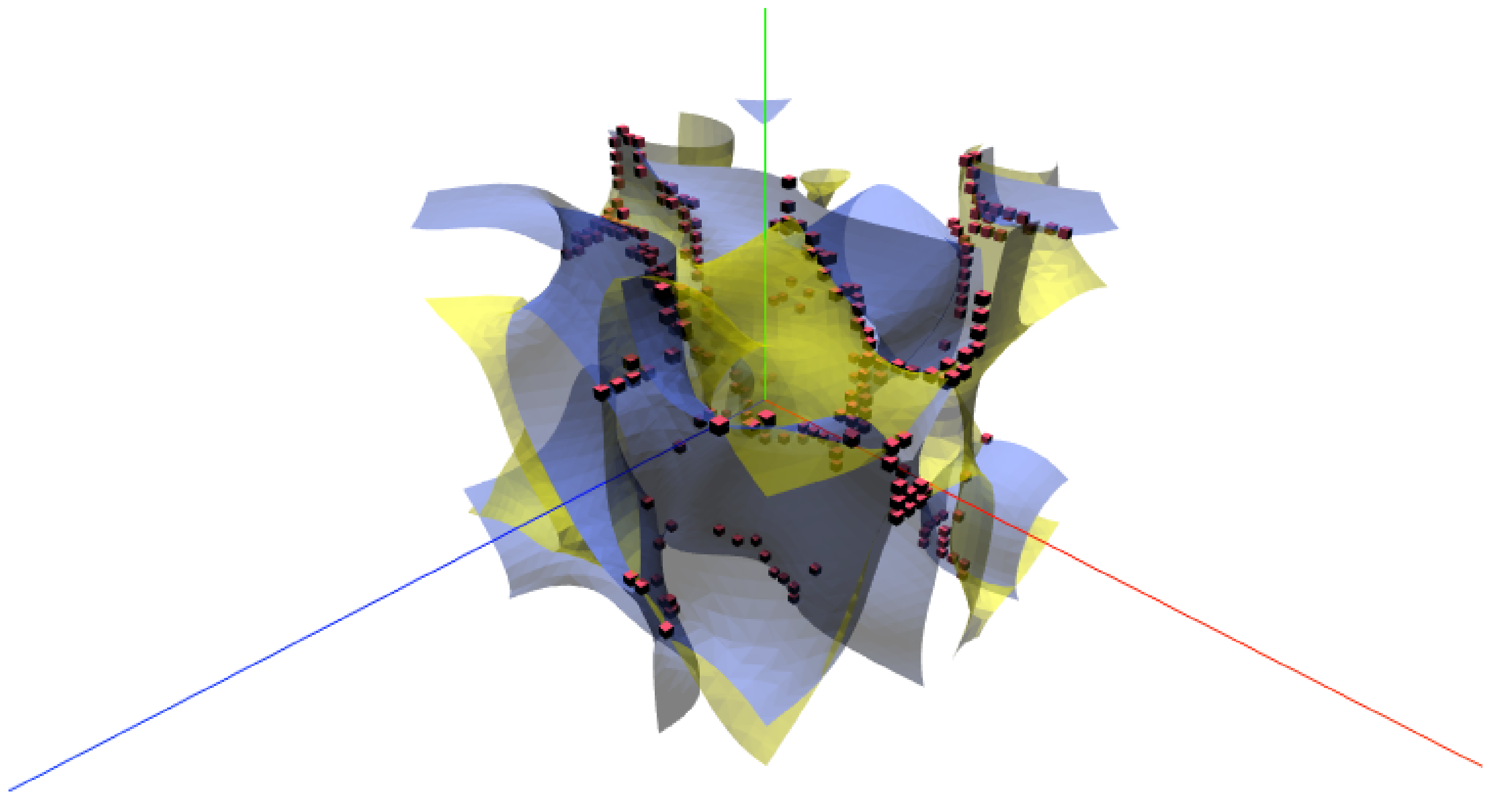
We now repeat the same process with one additional dimension. An array of arrays (2-D) can be represented as a matrix so we can display this as an image of size 64x64. At each of those 64x64 pixel we have a random value between -0.5 and +0.5 displayed here as colored circle (red is positive sign, blue regions indicate negative sign). Regions where the sign changes are displayed in intermediate yellow color.



Low-pass filter (iteration: 21)

Each time we low-pass filter the grid the structures become larger. Enable the display of the zero-crossings in the graph after a couple of smoothing steps. It is easy to see that an independently filtered second field would generate different zero-crossings. The intersection of those two independent sets of lines is a set of points. This pattern continues in dimensions greater than 2.

One more dimension! To not clutter our screen too much we will skip the display of the random values and only show the zero-crossings. We will do this twice on two independently generated and filtered white noise volumes. The zero-crossings are calculated using a marching cubes algorithm so we can visulize them as surfaces inside our data cube, one in blue for the second random field in yellow. In 3D this results for each random field in a 3-1=2D manifold (surface). Additionally to the two surfaces we also mark the locations in our 32x32x32 volume where the zero-crossings of both manifolds intersect, or more correcly, where they are both sufficiently close to zero as red cubes.



Low-pass filter (iteration: 68)

After many low-pass filter steps it becomes apparent that the intersections of the two 2-D manifolds are forming 3-1-1=1D manifolds - lines.

Summary: Each filtered random field of dimension N has a zero-crossing of dimension N-1. The intersection of two zero-crossings of dimension N-1 has dimension N-2.

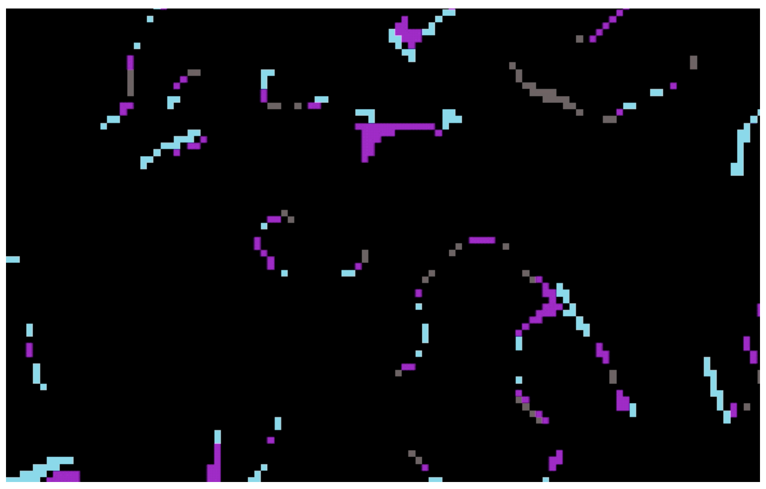
Ok, so how can we use this? We like to generate de novo data that in same respects resemble microvascular tissue. The zero-crossings of a low-pass filtered 3-D volume is a 2-D surface. The zero crossings of two low-pass filtered 3D volumes are objects of 2 dimensions less - so instead of surfaces we get 3 - 2 = 1-D lines. Can we interpret those as standins for blood vessels? Here an short movie of what these lines in 3D can look like if we increase the resolution of the volume.

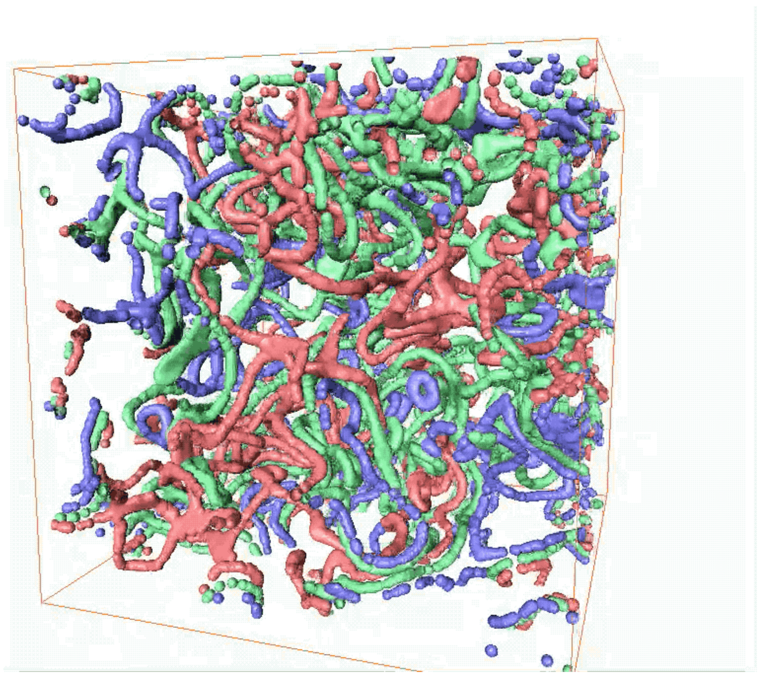
Link to the Movie

This volume was generated using the FakeBloodVessel program that implements this procedure for 3D data. The output are either labelled volumes (class 0: background, class 1: line-like structures) or a smoothed label field that simulates a blurring image aquisition.

There is more structure that we can get from this setup. The space in-between the intersecting zero-crossings can be described by the pair of signs of the low-pass filtered random fields. There are four possible combinations of signs ++, +-, -+, and -- so we get a random assortment of void spaces that will never intersect with our lines. Labels for these *void* regions are created by the application FakeBloodVessels using '-w'. The argument for option '-w' specifies how far away from the lines the regions are placed. The space in between is considered 'empty' (class 0).

More? Yes, there is more, The model is also able to create independent vessel tress that are guaranteed to never intersect with each other. Those *hugging* lines are generated by varying the zero of our zero-crossings. For the same random fields intersecting iso-surfaces at non-zero-crossings will form an independent vessel-like network that is geometrically close to the established network. Similar to broncial airways, arteries and veines? Here is what those independently generated sets of vessel like objects look like.





Ok, what next? We can now create de novo volumes that are filled with non-intersecting regions resembling blob-like areas (4 classes) and sets of non-intersecting vessel like structures. A generated volumes are based on independent white-noise. Given the resolution of our seed value this creates 4,294,967,295 different volumes with well specified features. This abundance of data is obviously well suited for the initial training of deep neural network models.

One might argue that data augmention can also be used to generate variety from a more limited set of real world dataset, which might be seen in contrast to the use of artificial data like the once presented here. But recent findings on adversarial noise attacks and data privacy breaches linked to overlearning stress the fact that data augmentation needs to be well designed. Here an example. Given a limited set of 2-D training data for deep neural networks in a classification task we can introduce data augmention to increase a limited pool of real-world data. Such data augmentation might mirror the data in 2 dimensions, rescale each image randomly and add some noise. In the worst case this results in a low number of original data being duplicated under the data augmentation transforms. None of those transforms changes the high-frequency noise pattern of the images. As a black-box method the deep neural network can therefore memorize the low number of original dataset by their noise pattern. If one of the limited noise pattern is detected the class can be assigned trivially. This will produce a network with a low training error. The validation error will be high - if the data is split before the augmentation step, which further reduces amount of data available for training and the danger of overlearning.

We suggest here that the benefits of purely artifical training data might outweight the draw-backs of a missmatch of model and reality. The benefits of using well-controlled artificial data for data privacy are obvious as black-box tests will not be able to identify particular individuals of our training set.

Notes

We call it low-pass filtering, why not band-pass filtered white noise? Oh, so yeah, we start with all kind of spatial frequencies in white noise, but by averaging we are retaining only the low-frequency variations. Each time we average we loose some of the higher-frequencies. .

Implementing a low-pass filter using averaging on a 6-neighbors 3-D stencil and applying it multiple times is not a good idea in practice. Here on this web-page we do this as an illustration only. It tends to still show the features discussed. The FakeBloodVessel application is using larger filter kernls to reduce the tendency of the smoothed data to be less smooth in diagonal directions. Applying the smoothing operation once in frequency space is another (future) todo.