# Instant image denoising: FAQ using ML models (Noise2Noise/DnCNN) and file conversions in ImageJ:

## Question 1: How to change the image denoising output to same as input image?

Ans: For accurate results (finer precision), the image denoising plugin results denoising images in 32-bit format always.

Example:

A picture containing text

Description automatically generated

Left: input noisy image, 8-bit image, right: denoised image, 32-bit image. To convert to input image format of 8-bit image: do the following conversions

**Edit -> Options -> Conversions: disable the “Scale when converting”**

Graphical user interface, text, application, chat or text message

Description automatically generated

Next, change the image format from the 32-bit to input format (here it is 8-bit image)

**Image -> Type -> 8-bit (from 32-bit of denoised image)**

after conversion:

A picture containing text

Description automatically generated

Left: input noisy image, 8-bit image, right: denoised image of 8-bit image which matches with the input image format. Let’s check the histogram: (before and after conversion)

Graphical user interface, text, application

Description automatically generated

Left: denoised image histogram (on 32-bit image), right: denoised image histogram (on 8-bit image)

After conversion from 32-bit to 8-bit, pixel values are integers, hence small difference in mean and standard deviation. Reference: <https://imagej.nih.gov/ij/docs/user-guide.pdf> (Section: 28)

## Question 2: How to perform image denoising with 3D stacks?

Answer:

Load the 3D image -> run Noise2Noise plugin, some examples and more details are provided here:

<https://github.com/ND-HowardGroup/Instant-Image-Denoising/tree/master/Plugins/Test_images/3D_images>

## Question 3: How to perform these ML-based image denoising models for color images?

Answer:

Our ML models (Noise2Noise and DnCNN) are trained with single channel (grayscale) images since the color image is the stack of multi-channel images (typically R, G and B channels). In simple terms, during training model, out training dataset dimension is (*NxCXHxW*) where *N*: batch size/batchwise operation, *C*=1 (single channel), *H* and *W* are height and width of the images respectively. Hence, the during the inference/evolution time, our models expect to have single channel image with batch of multiple channels.

To mitigate this problem, convert the multiple-channel image into stack of multiple images (stack) of single channel images.

In ImageJ for a single RGB image -> stack of single channel images:

Image -> Type -> RGB stack

Example:

Color image: left, right: channel1, channel2 and channel3 respectively. sample: BPAE sample

A picture containing graphical user interface

Description automatically generatedGraphical user interface

Description automatically generated with medium confidenceA picture containing graphical user interface

Description automatically generated

now perform, image denoising on this 3D stack and apply to 8-bit image if required and finally convert this stack to RGB/color image.

After denoising:

A picture containing graphical user interface

Description automatically generated

left: noisy color image -> convert to noisy 3D stack, right: denoised on the noisy 3D stack.

## Question 4: How to perform these ML-based image denoising models for 4D or 5D stacks?

Answer:

*Solution 1*: Most of the ImageJ plugins are written for 2D and 3D images and for the higher dimensions, we can convert the hyper stack to stack with the imageJ plugins and apply the image denoising on the 3D stack and convert back to hyper stack.

*Solution 2*: save all the images to 2D images in a folder and use the following macro to run the image denoising plugin through all the images in the specified folder.

Macro for image denoising for images in a folder:

input\_path = getDirectory("image"); //current noisy images folder location

print("\\Clear"); //clear log window

print("input\_path: ",input\_path);

filelist = getFileList(input\_path); //file list within this folder

for (i=0;i<filelist.length;i++)

{

print(filelist[i]); //present file name

run("Close All"); //close all open files/results

run("Clear Results"); //clear all results

open(input\_path + filelist[i]); //open ith image

run("Noise2Noise Denoising"); // run Noise2Noise model

setOption("ScaleConversions", False);

run("8-bit");

saveAs("Tiff","new\_+denoised\_"+filelist[i]); //save the denoised image

run("Close All"); //close input and denoised images

wait(2000); //wai 2sec (2000 ms) to clear the process

}

Reference:

<https://imagejdocu.tudor.lu/gui/image/hyperstacks>

## Question 5: List of existing image denoising methods as ImageJ plugins?

Answer:

* Traditional image denoising plugins (non ML methods):

1. ROF denoising: <https://javadoc.scijava.org/Fiji/fiji/denoise/ROF_Denoise.html>
2. Non-local means (NLM) denoising: <https://imagej.net/plugins/non-local-means-denoise>
3. Pure Denoise: <http://bigwww.epfl.ch/algorithms/denoise/>
4. Wavelet denoising: <https://imagej.net/plugins/wavelet-denoise>
5. Candle-J denoising: <https://github.com/haiderriazkhan/CANDLE-J>

* ML-based image denoising plugins: (ML methods)

1. CARE: <https://imagej.net/plugins/care> (supervised)
2. Noise2Void (N2V): <https://imagej.net/plugins/n2v> (self-supervised)
3. PN2V (probabilistic noise2void): <https://github.com/juglab/pn2v> (self-supervised with structured noise)
4. Denoiseg (denoising + segmentation): <https://imagej.net/plugins/denoiseg> (denoising and segmentation)

## Question 6: Comparison of existing ML models for image denoising methods/plugins?

Answer:

Noise2Noise ML based image denoising outperforms compare to the existing ML-base image denoising methods and better than conventional image denoising methods like block-matching 3D method (BM3D), non-local means (NLM), and mean-filtering/median filtering.

From [this reference](https://arxiv.org/pdf/1811.10980.pdf), (see Figure 7) the Noise2Noise/DnCNN models perform better than Noise2Void method and this is due to the fact that Noise2Noise was trained by using another noisy image within the same FOV whereas for the Noise2void was trained with only input noisy image within the same FOV.

Graphical user interface

Description automatically generated with medium confidence

From this figure, the traditional method refers to “Content-Aware image restoration” ([CARE](https://www.biorxiv.org/content/10.1101/236463v1.full.pdf)) image denoising method. Noise2Noise and DnCNN methods are performing better than other image denoising methods. For simplicity, the comparison is given here for the target image during training process used by the ML image denoising methods.

|  |  |  |  |
| --- | --- | --- | --- |
| **ML Method** | **Target image during training** | **Comments** | **Accuracy** |
| CARE (traditional from reference) | Clean image | Clean image/high SNR image requires high dosage/long integration time imaging | High |
| Noise2Noise | Another noisy image in the same FOV | Requires only another noisy image within FOV (fast imaging) | Very high |
| DnCNN | Clean image | Clean image/high SNR image requires high dosage/long integration time imaging | High |
| Noise2Void | Only noisy image (input) | No-need of any additional image as a target. | Moderate/low for faster training process |

## Question 7: Limitations of our demonstrated models?

Answer:

1. Memory limitation of ImageJ for processing of multiple slices: please check the size of the noisy image and if require process the image denoising in parts using method2 as mentioned in the previous question.
2. Noise2Noise ML method expects input will be multiple of 32x32 in size due to the max-pool layers. Hence divide the image into parts that are multiple of 32x32 before processing through Noise2Noise image denoising plugin.
   1. Methods: use the divide the into slices that are multiple of 32x32. (overlapping slices)
   2. if the input size is less than 32x32, image resize function is performed before applying the image denoising and restore back to original image dimensions.

## Question 8: Macro for image denoising for images in a given folder?

Answer:

input\_path = getDirectory("image"); //current noisy images folder location

print("\\Clear"); //clear log window

print("input\_path: ",input\_path);

filelist = getFileList(input\_path); //file list within this folder

for (i=0;i<filelist.length;i++)

{

print(filelist[i]); //present file name

run("Close All"); //close all open files/results

run("Clear Results"); //clear all results

open(input\_path + filelist[i]); //open ith image

run("Noise2Noise Denoising"); // run Noise2Noise model

setOption("ScaleConversions", False);

run("8-bit");

saveAs("Tiff","new\_+denoised\_"+filelist[i]); //save the denoised image

run("Close All"); //close input and denoised images

wait(2000); //wai 2sec (2000 ms) to clear the process

}

## Question 9: Other microscopy image datasets that includes noisy images for comparison?

Please refer to our FMD dataset which includes multiple fluorescence samples captured using various microscopy systems of BPAE samples, *in vivo* zebrafish, and mice brain images (noisy and ground truth images).

Link: <https://curate.nd.edu/show/f4752f78z6t>

For some more dataset, please refer to W2S image dataset which includes noisy images and corresponding ground-truth images (by taking an average of 400 images within the same FOV).

Link: <https://github.com/IVRL/w2s>

## Question 10: Where are the results/code for this paper including supplementary data information and imageJ plugins?

## Please refer to our GitHub repository: <https://github.com/ND-HowardGroup/Instant-Image-Denoising>

### **References:** Provided at the respective sections.

## Copyright information:

### **Author: Varun Mannam**

### Department of Electrical Engineering:

### University of Notre Dame, IN, USA, 46556

Email: [vmannam@nd.edu](mailto:vmannam@nd.edu)

GitHub: <https://github.com/ND-HowardGroup/Instant-Image-Denoising>