Image Processing and Analysis

Image denoising using deep learning techniques

Oge Marques, PhD

Professor

College of Engineering and Computer Science

FAU

FLORIDA ATLANTIC UNIVERSITY

Outline

1. Noise and noise models
2. Classical image denoising techniques (i.e., *before* the deep learning era)
3. Image denoising using MATLAB
4. Image denoising using deep learning in MATLAB
5. Hands-on assignment

What will we learn?

- What is noise (in the context of image processing) and how can it be modeled?
- What are the main types of noise that may affect an image?
- Which noise removal techniques are typically used in image processing?
- Which denoising techniques are available in MATLAB?
- Which deep learning architectures can be used for image denoising?
- How to implement image denoising using deep learning techniques in MATLAB?

Noise and Noise Models


Original
Denoised

What is image noise?

- Image noise is random variation of brightness or color information in images.
  - It is usually an aspect of electronic noise, which can be produced by the image sensor and circuitry of a scanner or digital camera.
  - It is an undesirable by-product of image capture that obscures the desired information.
- In the context of signal (and image) processing, “noise” can be interpreted as “unwanted signal”.

Source: Wikipedia https://en.wikipedia.org/wiki/Image_noise

Types of noise

- Gaussian noise
- Salt-and-pepper noise
- Shot noise
- Quantization noise (uniform noise)
- Film grain
- Periodic noise

Source: Wikipedia https://en.wikipedia.org/wiki/Image_noise

Gaussian noise

- Gaussian noise, named after Carl Friedrich Gauss, is statistical noise having a probability density function (PDF) equal to that of the normal distribution, which is also known as the Gaussian distribution.
- Image sensors have inherent noise due to the level of illumination and their own temperature, and the electronic circuits connected to the sensor inject their own share of electronic circuit noise.
- A typical model of image / sensor noise is Gaussian, additive, independent at each pixel, and independent of the signal intensity, caused primarily by Johnson–Nyquist noise (thermal noise).

Source: Wikipedia https://en.wikipedia.org/wiki/Image_noise and https://en.wikipedia.org/wiki/Gaussian_noise

Salt-and-pepper noise

- Sometimes referred to as "impulsive”, speckle, or spike noise, it appears in the form of dark pixels in bright regions and bright pixels in dark regions of an image.

- This type of noise can be caused by analog-to-digital converter errors, bit errors in transmission, among other reasons.

Source: Wikipedia https://en.wikipedia.org/wiki/Image_noise and https://en.wikipedia.org/wiki/Salt_and_pepper_noise

Shot noise

- Photon shot noise is the dominant noise in the brighter parts of an image from an image sensor is typically that caused by statistical quantum fluctuations, that is, variation in the number of photons sensed at a given exposure level.
- Shot noise has a root-mean-square value proportional to the square root of the image intensity, and the noises at different pixels are independent of one another.
- Shot noise follows a Poisson distribution, which except at very high intensity levels approximates a Gaussian distribution.

Source: Wikipedia https://en.wikipedia.org/wiki/Image_noise

Quantization noise

- The noise caused by quantizing the pixels of a sensed image to a number of discrete levels is known as quantization noise.

- It has an approximately uniform distribution.

Source: Wikipedia https://en.wikipedia.org/wiki/Image_noise

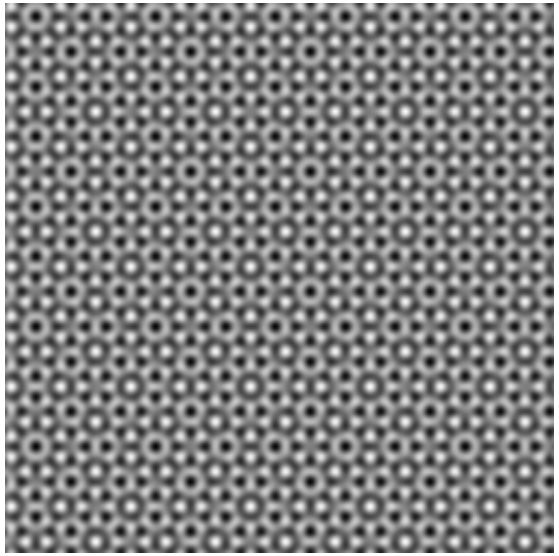
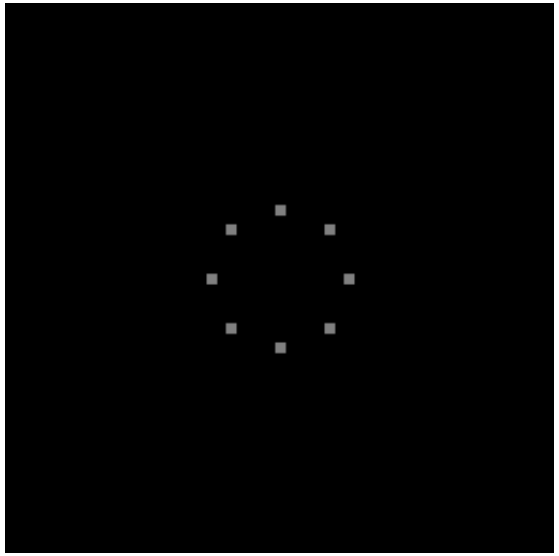
Film grain

- The grain of photographic film is a signal-dependent noise, with similar statistical distribution to shot noise.
- If film grains are uniformly distributed (equal number per area), and if each grain has an equal and independent probability of developing to a dark silver grain after absorbing photons, then the number of such dark grains in an area will be random with a binomial distribution.
- In areas where the probability is low, this distribution will be close to the classic Poisson distribution of shot noise.
- A simple Gaussian distribution is often used as an adequately accurate model.

Source: Wikipedia https://en.wikipedia.org/wiki/Image_noise

Periodic noise

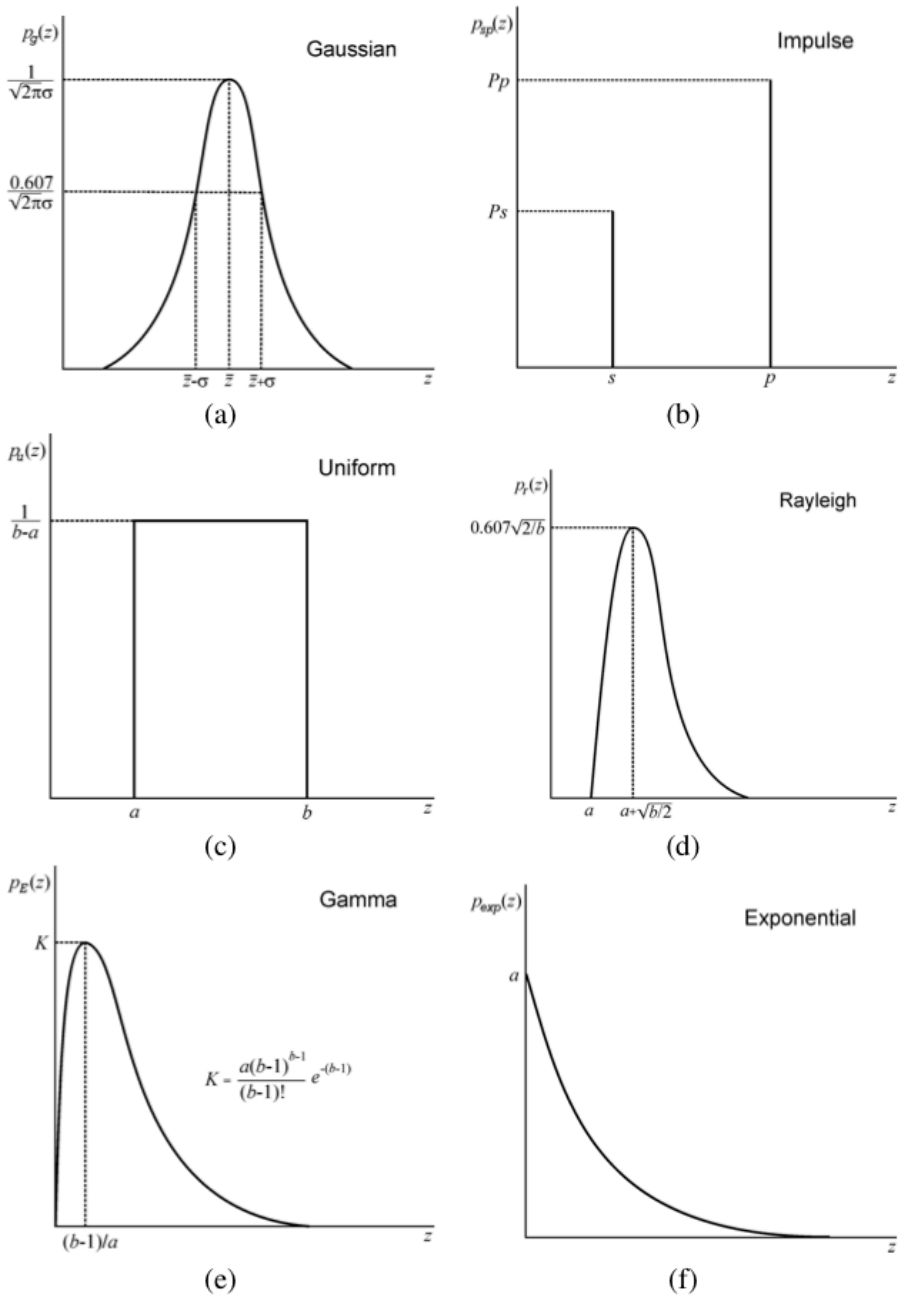
- Periodic noise
  - a type of noise that usually arises as a result of electrical or electromechanical interference during the image acquisition process.

Source: Wikipedia https://en.wikipedia.org/wiki/Image_noise

Selected noise models

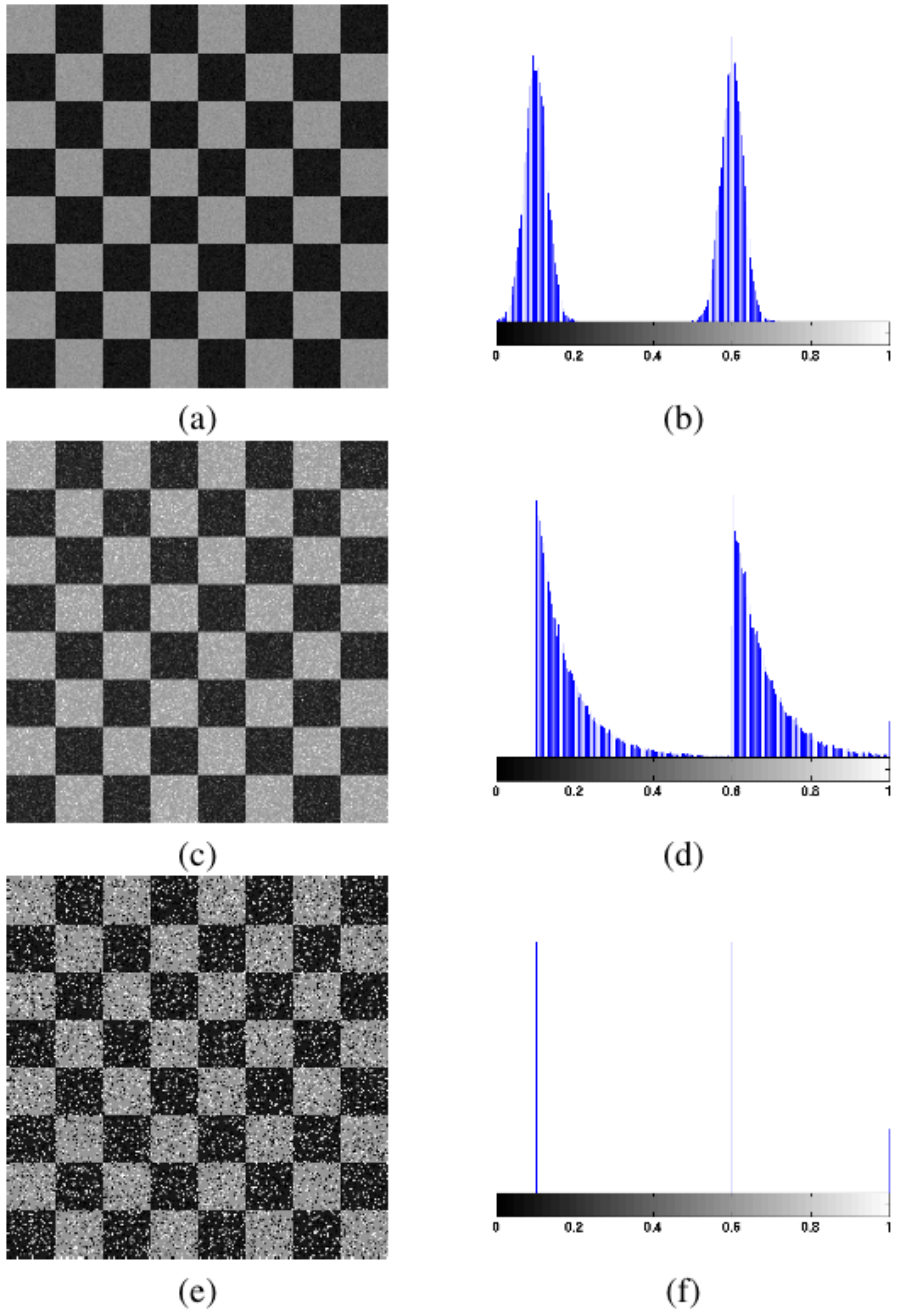
- Gaussian
- Impulse (salt-and-pepper)
- Uniform
- Rayleigh
- Gamma (Erlang)
- Exponential



Source: Marques (2011) “Practical Image and Video Processing Using MATLAB”, Wiley.

Effects of noise on images and histograms

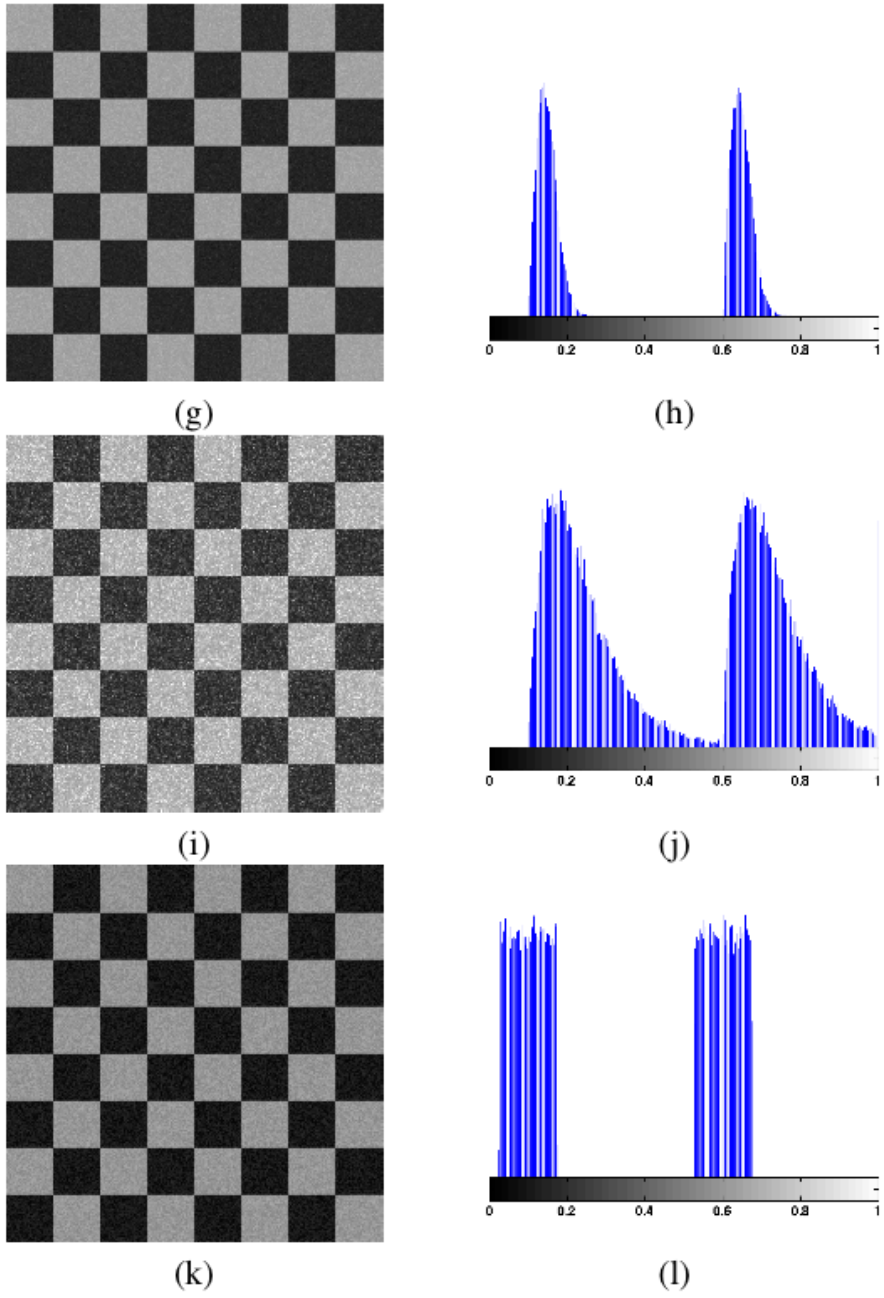
- Gaussian
- Exponential
- Impulse (salt-and-pepper)



Source: Marques (2011) “Practical Image and Video Processing Using MATLAB”, Wiley.

Effects of noise on images and histograms

- Rayleigh
- Gamma (Erlang)
- Uniform

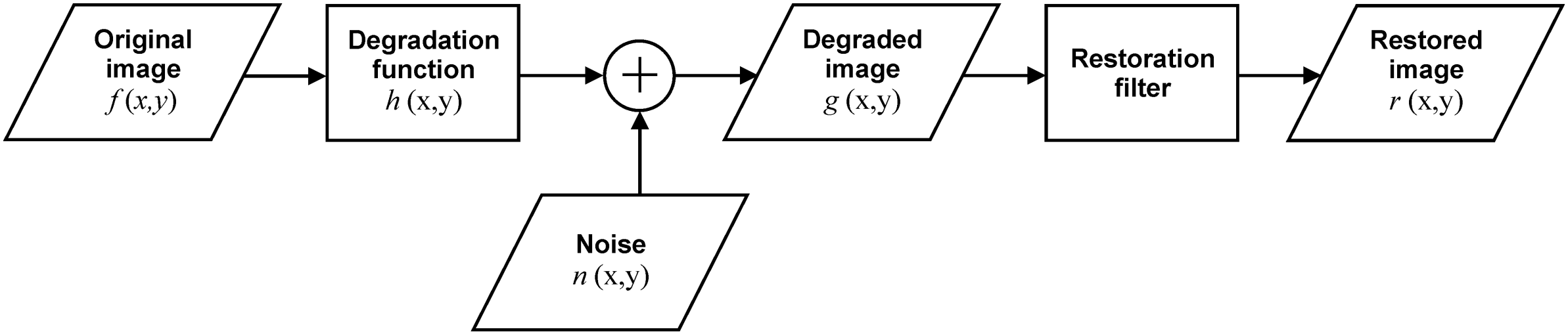


Source: Marques (2011) “Practical Image and Video Processing Using MATLAB”, Wiley.

Classical Image Denoising Techniques

(i.e., *before* the Deep Learning Era)

Modeling of the image degradation and restoration problem



Spatial domain

$$g(x,y) = f(x,y) * h(x,y) + n(x,y)$$

Frequency domain

$$G(u,v) = F(u,v)H(u,v) + N(u,v)$$

Source: Marques (2011) “Practical Image and Video Processing Using MATLAB”, Wiley.

Noise reduction – spatial domain

- Mean filters
  - Arithmetic
  - Geometric
  - Harmonic
  - Contra-harmonic
- Order-statistic (rank) filters
  - Median
  - Min
  - Max
  - Mid-point
- Alpha-trimmed mean filter

Source: Marques (2011) “Practical Image and Video Processing Using MATLAB”, Wiley.

Noise reduction – Mean filters

- Arithmetic mean filter (averaging filter)

$$\hat{f}(x,y) = \frac{1}{mn} \sum_{(r,c)\in W} g(r,c)$$

  - Causes a certain amount of blurring (proportional to the window size) to the image, thereby reducing the effects of noise.
  - Can be used to reduce noise of different types, but works best for Gaussian, uniform, or Erlang noise.

Source: Marques (2011) “Practical Image and Video Processing Using MATLAB”, Wiley.

Noise reduction – Mean filters

- Geometric mean filter

$$\hat{f}(x,y) = \left[\prod_{(r,c)\in W} g(r,c) \right]^{\frac{1}{mn}}$$

  - a variation of the arithmetic mean filter
  - primarily used on images with Gaussian noise
  - retains image detail better than the arithmetic mean filter

Source: Marques (2011) “Practical Image and Video Processing Using MATLAB”, Wiley.

Noise reduction – Mean filters

- Harmonic mean filter

$$\hat{f}(x,y) = \frac{mn}{\sum_{(r,c)\in W} \frac{1}{g(r,c)}}$$

  - another variation of the arithmetic mean filter
  - useful for images with Gaussian or salt noise
  - black pixels (pepper noise) are not filtered

Source: Marques (2011) “Practical Image and Video Processing Using MATLAB”, Wiley.

Noise reduction – Order-statistic filters

- Median filter

$$\hat{f}(x,y) = \text{median}\,\{g(r,c)|(r,c) \in W\}$$

  - **Most popular and useful of the rank filters**.
  - It works by selecting the middle pixel value from the ordered set of values within the $m \times n$ neighborhood (W) around the reference pixel.
    - If mn is an even number (not common), the arithmetic average of the two values closest to the middle of the ordered set is used instead.
  - Many variants, extensions, and optimized implementations in the literature.

Source: Marques (2011) “Practical Image and Video Processing Using MATLAB”, Wiley.

Noise reduction – Order-statistic filters

- Min and Max filters
  - Min: replaces the reference pixel with the lowest value of the ordered set

$$\hat{f}(x,y) = \min\{g(r,c) | (r,c) \in W\}$$

  - Max: replaces the reference pixel with the highest value of the ordered set

$$\hat{f}(x,y) = \max\{g(r,c) | (r,c) \in W\}$$

Source: Marques (2011) “Practical Image and Video Processing Using MATLAB”, Wiley.

Noise reduction – Order-statistic filters

- Midpoint filter

$$\hat{f}(x,y) = \frac{1}{2}\left[\max\left\{g(r,c)|(r,c) \in W\right\} + \min\left\{g(r,c)|(r,c) \in W\right\}\right]$$

  - Calculates the average of the highest and lowest pixel values within a window, thereby combining order statistics and averaging into one filter.
  - Used to reduce Gaussian and uniform noise.

Source: Marques (2011) “Practical Image and Video Processing Using MATLAB”, Wiley.

Noise reduction – Alpha-trimmed mean filter

- Alpha-trimmed mean filter

$$\hat{f}(x,y) = \frac{1}{mn - 2D} \sum_{(r,c) \in W} g(r,c)$$

  - Uses another combination of order statistics and averaging, in this case an average of the pixel values closest to the median, after the *D* lowest and the *D* highest values in an ordered set have been excluded.

  - Rationale: to allow its user to control its behavior by specifying the parameter *D*:
    - for *D* = 0: the filter behaves like a regular arithmetic mean filter;
    - for *D* =(mn-1)/2: it is equivalent to the median filter.

  - Used in cases where the image has been corrupted by more than one type of noise, e.g., salt-and-pepper and Gaussian.

Source: Marques (2011) “Practical Image and Video Processing Using MATLAB”, Wiley.

Noise reduction – frequency domain

- Bandreject filter
  - attenuates frequency components within a certain range (the *stopband* of the filter) while leaving all other frequency components untouched (or amplifying them by a certain gain).

  - Ideal bandreject filter

$$H_{br}^{i}(u,v) = \begin{cases} 1 & \text{if } D(u,v) < D_0 - \frac{W}{2} \\ 0 & \text{if } D_0 - \frac{W}{2} \leq D(u,v) \leq D_0 + \frac{W}{2} \\ 1 & \text{if } D(u,v) > D_0 + \frac{W}{2} \end{cases}$$

Source: Marques (2011) “Practical Image and Video Processing Using MATLAB”, Wiley.

Noise reduction – frequency domain

- Bandreject filter
  - Butterworth bandreject filter $$H_{br}^{b}(u,v) = \frac{1}{1+\left[\frac{D(u,v)W}{D^2(u,v)-D_0^2}\right]^{2n}}$$
  - Gaussian bandreject filter $$H_{br}^{g}(u,v) = 1 - e^{-\frac{1}{2}\left[\frac{D^2(u,v)-D_0^2}{D(u,v)W}\right]^2}$$

Source: Marques (2011) “Practical Image and Video Processing Using MATLAB”, Wiley.

Noise reduction – frequency domain

- Bandpass filter
  - Allows certain frequencies (within its *passband*) to be preserved while attenuating all others.
  - It is, in effect, the opposite of a bandreject filter:

$$H_{bp}(u,v) = 1 - H_{br}(u,v)$$

Source: Marques (2011) “Practical Image and Video Processing Using MATLAB”, Wiley.

Noise reduction – frequency domain

- Bandpass filter (BPF)
  - Ideal

$$H_{bp}^{i}(u,v) = \begin{cases} 0 & \text{if } D(u,v) < D_0 - \frac{W}{2} \\ 1 & \text{if } D_0 - \frac{W}{2} \leq D(u,v) \leq D_0 + \frac{W}{2} \\ 0 & \text{if } D(u,v) > D_0 + \frac{W}{2} \end{cases}$$

  - Butterworth

$$H_{bp}^{b}(u,v) = \frac{\left[\frac{D(u,v)W}{D^2(u,v)-D_0^2}\right]^{2n}}{1+\left[\frac{D(u,v)W}{D^2(u,v)-D_0^2}\right]^{2n}}$$

  - Gaussian

$$H_{bp}^{g}(u,v) = e^{-\frac{1}{2}\left[\frac{D^2(u,v)-D_0^2}{D(u,v)W}\right]^2}$$

Source: Marques (2011) “Practical Image and Video Processing Using MATLAB”, Wiley.

Noise reduction – frequency domain

- Notch filter
  - Special kind of frequency domain filter that attenuates (or allows) frequencies within a *neighborhood* around a center frequency.
    - Due to the symmetry property of the Fourier transform, the spectrum of notch filters shows symmetric pairs around the origin (except for a notch filter located at the origin, of course).

Source: Marques (2011) “Practical Image and Video Processing Using MATLAB”, Wiley.

Noise reduction – frequency domain

- Ideal notch filter

$$H_{nr}^{i}(u,v) = \begin{cases} 0 & \text{if } D_1(u,v) < D_0 \text{ or } D_2(u,v) < D_0 \\ 1 & \text{otherwise} \end{cases}$$

where:

$$D_1(u,v) = \left[(u - M/2 - u_0)^2 + (v - N/2 - v_0)^2)\right]^{1/2}$$

and

$$D_2(u,v) = \left[(u - M/2 + u_0)^2 + (v - N/2 + v_0)^2)\right]^{1/2}$$

Source: Marques (2011) “Practical Image and Video Processing Using MATLAB”, Wiley.

Noise reduction – frequency domain

- Butterworth notch filter

$$H_{nr}^{b}(u,v) = \frac{1}{1 + \left[\frac{D_0^2}{D_1(u,v)D_2(u,v)}\right]^n}$$

where:

$$D_1(u,v) = \left[(u - M/2 - u_0)^2 + (v - N/2 - v_0)^2)\right]^{1/2}$$

and

$$D_2(u,v) = \left[(u - M/2 + u_0)^2 + (v - N/2 + v_0)^2)\right]^{1/2}$$

Source: Marques (2011) “Practical Image and Video Processing Using MATLAB”, Wiley.

Noise reduction – frequency domain

- Gaussian notch filter

$$H_{nr}^{g}(u,v) = 1 - e^{-\frac{1}{2}\left[\frac{D_1(u,v)D_2(u,v)}{D_0^2}\right]}$$

where:

$$D_1(u,v) = \left[(u - M/2 - u_0)^2 + (v - N/2 - v_0)^2)\right]^{1/2}$$

and

$$D_2(u,v) = \left[(u - M/2 + u_0)^2 + (v - N/2 + v_0)^2)\right]^{1/2}$$

Source: Marques (2011) “Practical Image and Video Processing Using MATLAB”, Wiley.

Noise reduction – frequency domain

To convert a notch filter that rejects certain frequencies into one that allows those same frequencies to pass one has to simply compute:

$$H_{np}(u,v) = 1 - H_{nr}(u,v)$$

where $H_{np}(u, v)$ is the transfer function of the notch pass filter that corresponds to the notch reject filter whose transfer function is $H_{nr}(u, v)$.

Source: Marques (2011) “Practical Image and Video Processing Using MATLAB”, Wiley.

Image denoising using MATLAB

Image denoising using MATLAB: useful functions

- `imnoise`
  - https://www.mathworks.com/help/images/ref/imnoise.html
  - Used to add noise to an image
  - Noise types: Gaussian, Poisson, salt and pepper, multiplicative (speckle)

- `imfilter`
  - https://www.mathworks.com/help/images/ref/imfilter.html
  - N-D filtering of multidimensional images with user-defined filter mask (kernel), which can be defined manually or with the help of the `fspecial` function

- `fspecial`
  - https://www.mathworks.com/help/images/ref/fspecial.html
  - Predefined 2D filter kernels, e.g.: average, disk, gaussian

Source: https://www.mathworks.com/help/images/noise-removal.html

Image denoising using MATLAB: useful functions

- `medfilt2`
  - https://www.mathworks.com/help/images/ref/medfilt2.html
  - Performs median filtering on an image
- `ordfilt2`
  - https://www.mathworks.com/help/images/ref/ordfilt2.html
  - Performs 2D order-statistic filtering (e.g., max, min) on an image
- `wiener2`
  - https://www.mathworks.com/help/images/ref/wiener2.html
  - Applies a pixel-wise adaptive low-pass Wiener filter to an image

Source: https://www.mathworks.com/help/images/noise-removal.html

Image denoising using MATLAB: useful functions

- `imdiffusefilt`
  - https://www.mathworks.com/help/images/ref/imdiffusefilt.html
  - Applies anisotropic diffusion filtering to an image
- `imnlmfilt`
  - hhttps://www.mathworks.com/help/images/ref/imnlmfilt.html
  - Applies a non-local means-based filter to an image
- `locallapfilt`
  - https://www.mathworks.com/help/images/ref/locallapfilt.html
  - Filters an image with an edge-aware, fast local Laplacian filter.

Source: https://www.mathworks.com/help/images/noise-removal.html

Image denoising using MATLAB: useful functions

- `imgaussfilt`
  - https://www.mathworks.com/help/images/ref/imgaussfilt.html
  - Filters image with a 2-D Gaussian smoothing kernel

- `imbilatfilt`
  - https://www.mathworks.com/help/images/ref/imbilatfilt.html
  - Applies an edge-preserving Gaussian bilateral filter to an image

Source: https://www.mathworks.com/help/images/noise-removal.html

Image denoising using deep learning in MATLAB

Image denoising using deep learning in MATLAB

3 main options:

1. Use the built-in pretrained denoising neural network, (*DnCNN*)
2. Train your own denoising CNN using built-in layers
3. Use an autoencoder.

Reference: https://www.mathworks.com/help/images/train-and-apply-denoising-neural-networks.html

Image denoising using deep learning in MATLAB

- OPTION 1 (Simplest and fastest solution): use the **built-in pretrained denoising neural network, *DnCNN***.

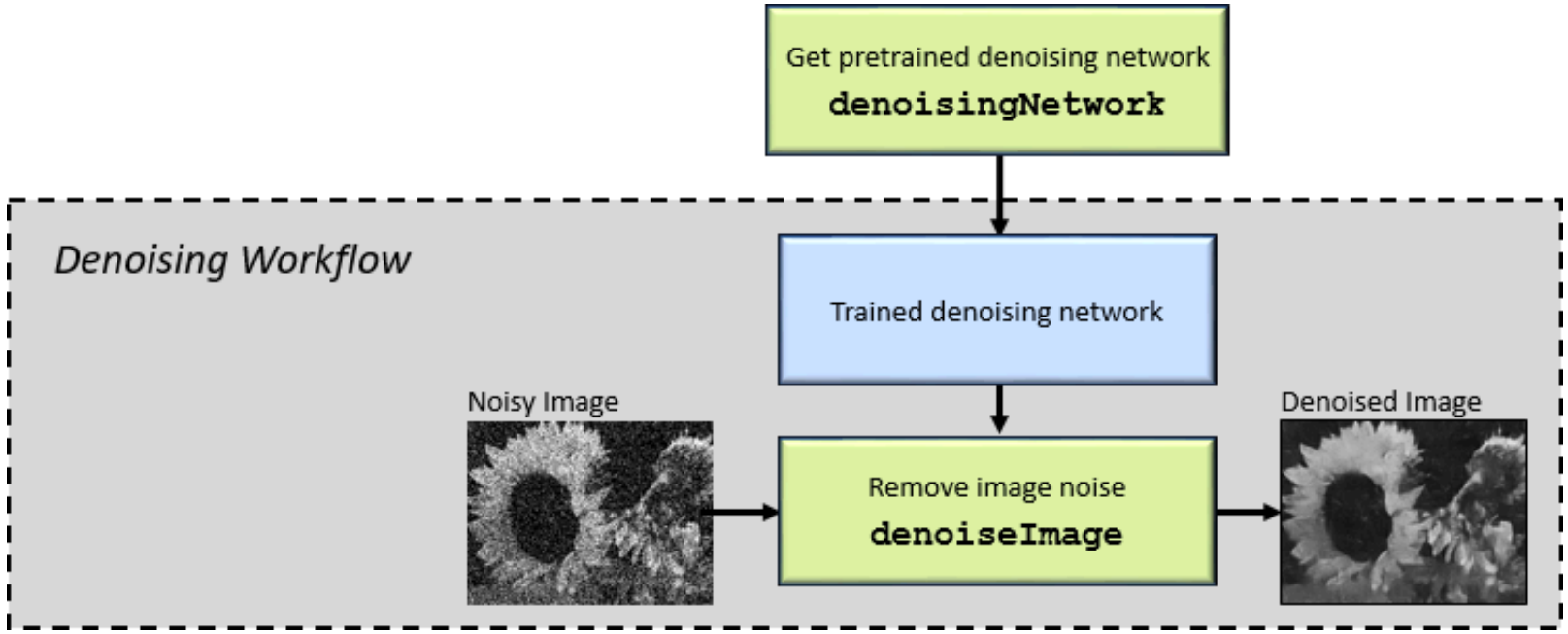
- Limitations:
  1. Noise removal works only with 2D monochrome (grayscale) images.
     - For images with multiple color channels, or 3D images, noise removal must be done by treating each channel or plane separately.

  2. The network recognizes only Gaussian noise, with a limited range of standard deviation.

Reference: https://www.mathworks.com/help/images/train-and-apply-denoising-neural-networks.html

Image denoising using pretrained **DnCNN** in MATLAB

- Steps:
  1. To load the pretrained DnCNN network, use the denoisingNetwork function.
  2. Pass the DnCNN network and a noisy 2D single-channel image to denoiseImage.



Reference: https://www.mathworks.com/help/images/train-and-apply-denoising-neural-networks.html

Image denoising using pretrained **DnCNN** in MATLAB

- Example code (for grayscale images):

Remove Image Noise Using Pretrained Neural Network

Load the pretrained denoising convolutional neural network, `'DnCNN'`.

```
net = denoisingNetwork('DnCNN');
```

Load a grayscale image into the workspace, then create a noisy version of the image.

```
I = imread('cameraman.tif');
noisyI = imnoise(I,'gaussian',0,0.01);
```

Display the two images as a montage.

```
montage({I,noisyI})
title('Original Image (Left) and Noisy Image (Right)')
```

Remove noise from the noisy image, then display the result.

```
denoisedI = denoiseImage(noisyI,net);
imshow(denoisedI)
title('Denoised Image')
```


Reference: https://www.mathworks.com/help/images/train-and-apply-denoising-neural-networks.html

Color image denoising using pretrained **DnCNN** in MATLAB

In practice, image color channels frequently have **correlated noise**.

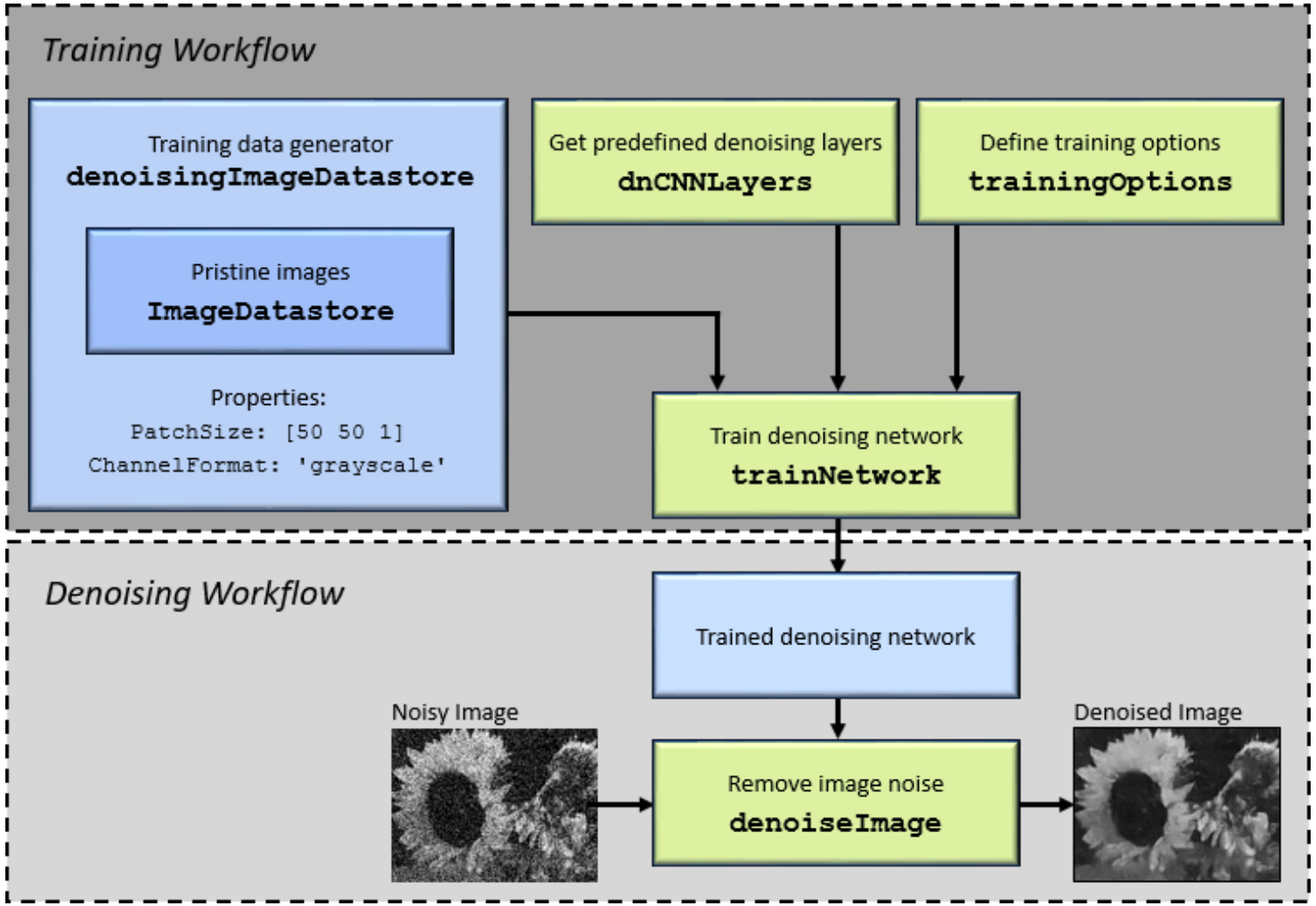
To remove correlated image noise, follow these steps:

1. Load color image into the workspace and convert the data to double.
2. Add zero-mean Gaussian white noise with a variance of 0.01 to the image.
3. Convert the RGB image to a color space with a luminance channel, such as the L*a*b* color space
4. Load the pretrained DnCNN network.
5. Use the DnCNN network to remove noise from the luminance channel only.
6. Convert the denoised image back to the RGB color space.
7. Display images (original, noisy, denoised).
8. Compute figures of merit (PSNR, SSIM, etc.)

Reference: https://www.mathworks.com/help/images/remove-noise-from-color-image-using-pretrained-neural-network.html

Image denoising using deep learning in MATLAB

- OPTION 2: Build and train your own denoising neural network



Reference: https://www.mathworks.com/help/images/train-and-apply-denoising-neural-networks.html

Image denoising using deep learning in MATLAB

Building and (re-)training your own network

Steps:

1. Create an ImageDatastore object that stores pristine images.
2. Create a denoisingImageDatastore object that generates noisy training data from the pristine images.
3. Get the predefined denoising layers using the dnCNNLayers function.
4. Define training options using the trainingOptions function.
5. Train the network, specifying the denoising image datastore as the data source for trainNetwork.
6. After you have trained the network, pass the network and a noisy grayscale image to denoiseImage.

Reference: https://www.mathworks.com/help/images/train-and-apply-denoising-neural-networks.html

Image denoising using deep learning in MATLAB

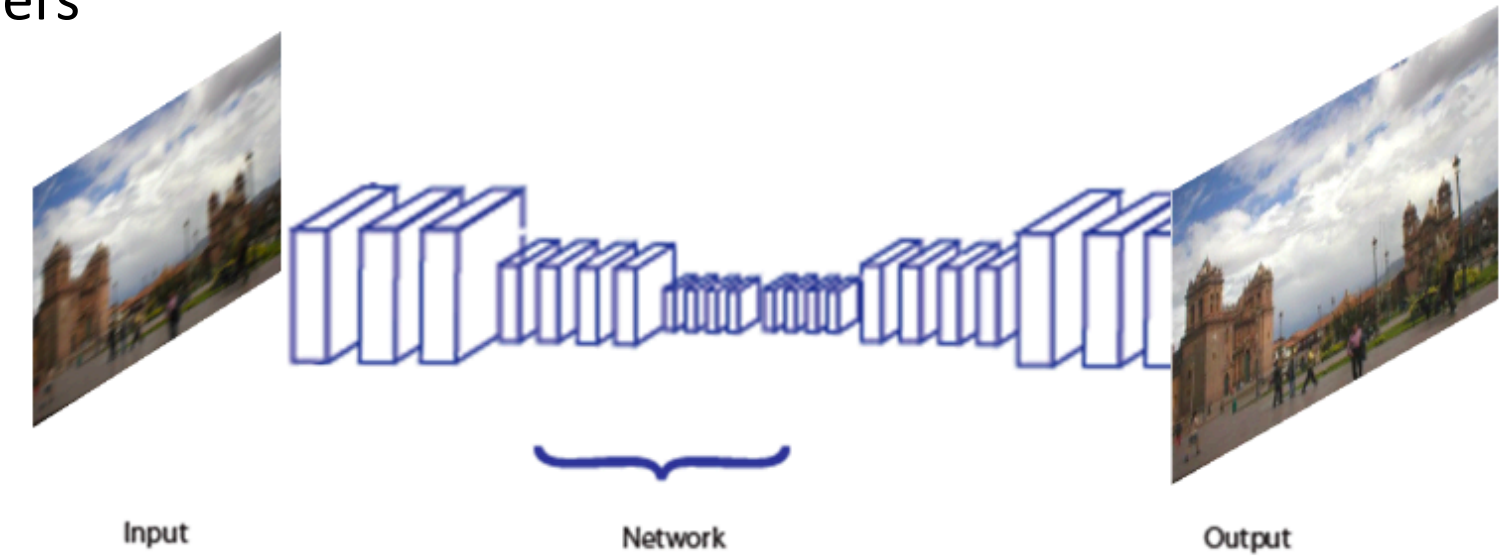
OPTION 3: Use a convolutional autoencoder.

- Convolutional autoencoders consist of two stages: an encoder and a decoder.
  - The encoder compresses the original input image into a latent representation that is smaller in width and height, but deeper in the sense that there are many feature maps per spatial location than the original input image.
    - The compressed latent representation loses some amount of spatial resolution in its ability to recover high frequency features in the original image, but it also learns to not include noisy artifacts in the encoding of the original image.
  - The decoder repeatedly upsamples the encoded signal to move it back to its original width, height, and number of channels.

Reference: https://www.mathworks.com/help/deeplearning/ug/image-to-image-regression-using-deep-learning.html

Convolutional autoencoders and image-to-image regression

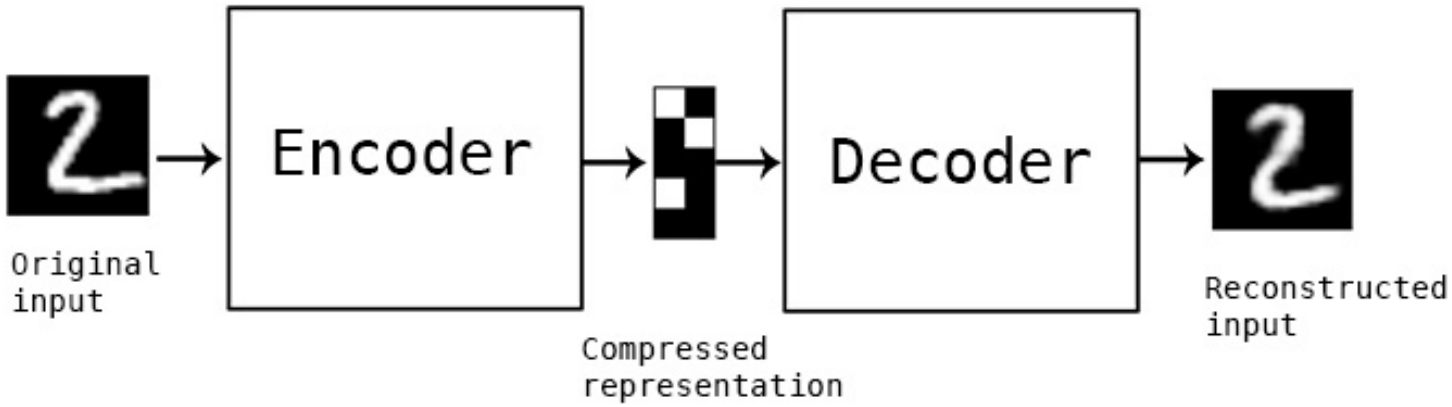
- Image-to-image regression: the process of estimating new values for each pixel of an image (provided as input to a deep neural network)
- Architectures include:
  - Convolutional autoencoders
  - U-nets
  - etc.
- Applications include:
  - Semantic image segmentation
  - **Denoising**
  - Colorization
  - etc.



Reference: https://blogs.mathworks.com/deep-learning/2019/02/21/image-to-image-regression/

Image denoising using deep learning in MATLAB

- Convolutional autoencoders are convenient for denoising operations!
  - Since the encoder removes noise (as part of its job in building a more compact representation of the input image), the decoded final image has fewer noise artifacts.

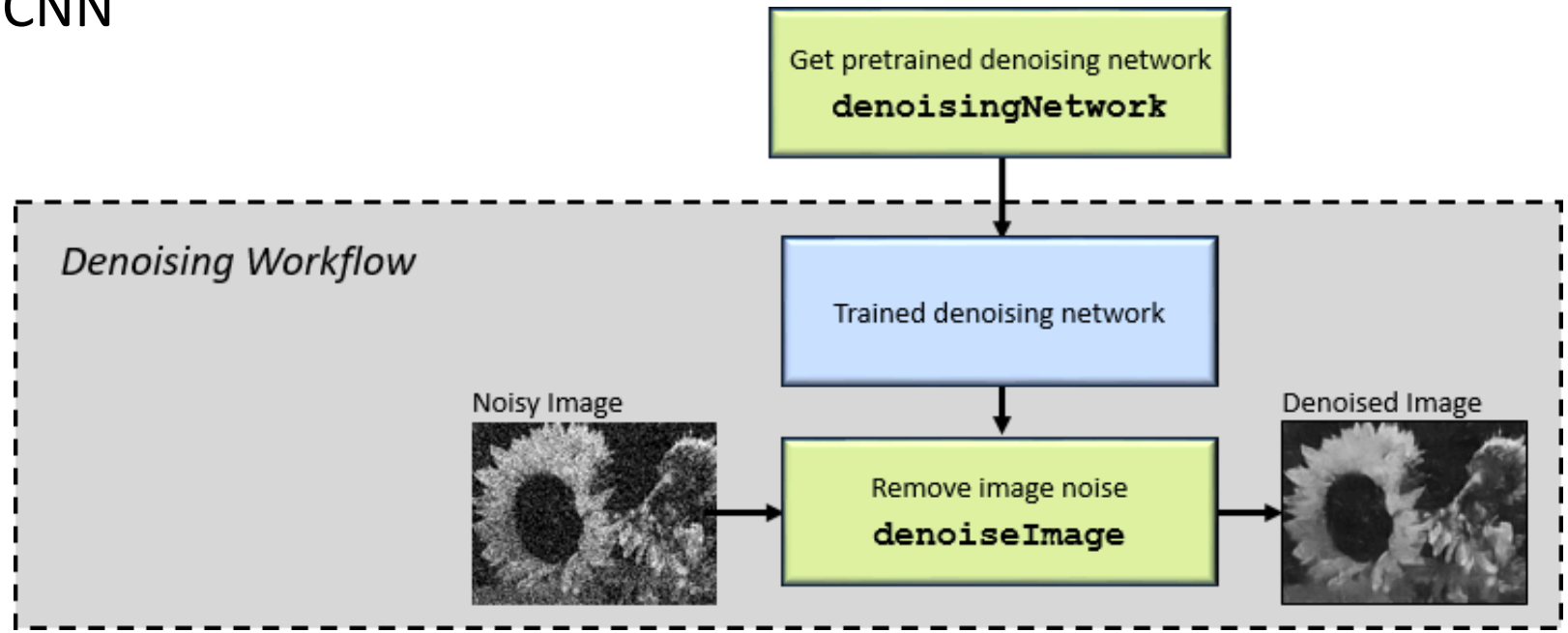


Reference: https://www.mathworks.com/help/deeplearning/ug/image-to-image-regression-using-deep-learning.html | Figure from: https://blog.keras.io/building-autoencoders-in-keras.html

Hands on!

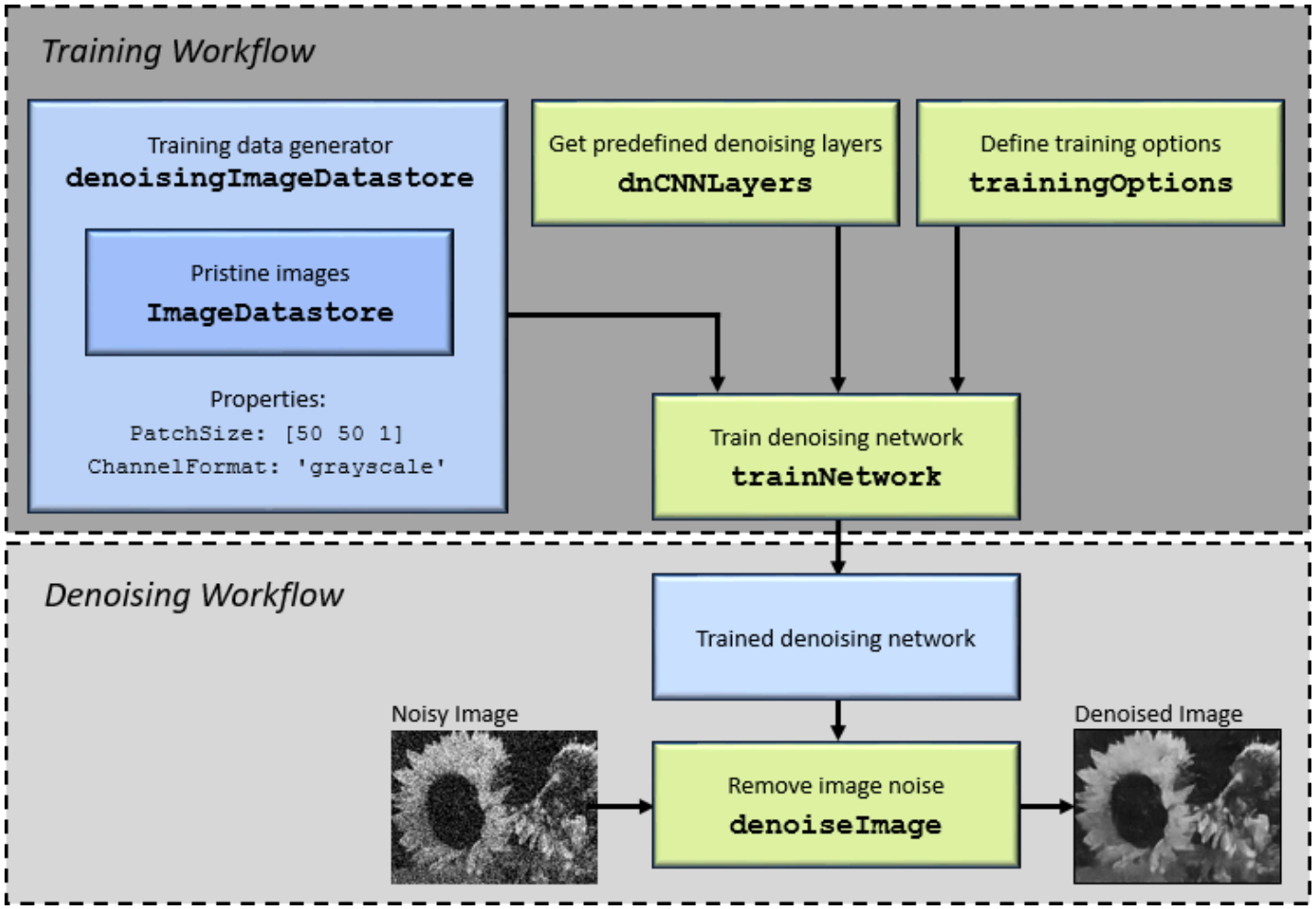
MATLAB Assignment:
Image denoising using deep learning (denoising_dl.mlx)

- Part 1:
  - Quick review of classical of image noise types and measures of image quality
- Part 2:
  - Simplest workflow for removing image noise from an individual image using a pre-trained denoising CNN



MATLAB Assignment:
Image denoising using deep learning (denoising_dl.mlx)

- Part 3:
  - Complete workflow for **training** a denoising CNN using a datastore of pristine images and eventually deploying the resulting model



MATLAB Assignment:
Image denoising using deep learning (denoising_dl.mlx)

- Part 4 (OPTIONAL):
  - Workflow for training a convolutional autoencoder for denoising purposes.