UNIVERSITY OF COPENHAGEN
FACULTY OF SCIENCE

Bachelor Thesis

Frederik Henriques Altmann
Gustav Emil Mark-Hansen

Effects of Demosaicing, Noise, and Image Processing on Mean Signal Reconstruction Error U-Net Performance

Date: June 21, 2024

Advisors: Ankit Kariryaa & Christian Igel

1. *Abstract*

A Colour Filter Array (CFA) extracts colour information from digital image sensors. The literature on learning from CFA data, usually focuses on specific downstream tasks, and lacks a systematic investigation of how image processing used to convert CFA data to red-green-blue (RGB) images affects subsequent deep learning processes.

We propose the following two signal denoising experiments;

1) Reconstructing images from noisy CFA data and RGB images, synthesized from the same data source, using common image processing settings. Our experiments show a significant difference, in the mean signal reconstruction error (MSRE) for denoising models trained on CFA, and RGB images processed using the adaptive-homogeneity directed (AHD) signal reconstruction (demosaicing) algorithm, the latter RGB models having higher MSRE.

2) Reconstructing images from CFA data and RGB images, that are synthesized from the same data source. The RGB images are processed with different settings to compare MSRE for different kinds of processing. Our experiments show lower MSRE for CFA models than compared to most RGB models. The RGB models, that are trained on images processed with more than just demosaicing, having higher MSRE.

Index terms - Colour Filter Array, Demosaicing, Signal Reconstruction, Deep Learning

Contents

2. *Introduction*

Digital photography often uses a single Colour Filter Array (CFA) coupled sensor. Such a sensor, samples the radiance of the scene imperfectly in different colour bands. To create a full colour image, the sampled signal is reconstructed. This reconstruction is generally optimized for human viewing. The non-linear processing associated with this reconstruction may reduce the informational content, leading to lower data fidelity in the context of machine learning (ML).

Many datasets widely used for ML, consist of images with red-blue-green (RGB) colour channels. These images are most often taken with digital cameras, motivating studying the way digital images are processed, and how this processing effects ML performance.

In this context, the hypothesis is; The mean signal reconstruction error (MSRE) when reconstructing an original signal, from a processed RGB image, is higher, than reconstructing from a CFA capture. The image processing might provide a local minima in terms of MSRE, but limit the potential of further processing to lower the MSRE.

To reconstruct the signal, a 5-layer U-Net deep learning model is used. It's used since the architecture is widely studied and signal reconstruction models based on U-Net performs well in literature [1].

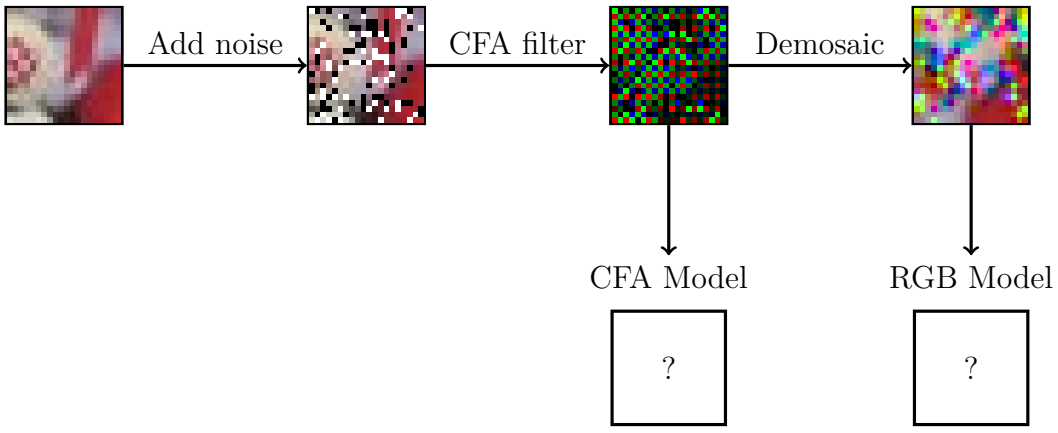

Figure 2.1: Experiment setup

The scenarios investigated in the study are;

1) Training denoising models on synthetic noisy CFA data, and comparing to models trained on RGB images reconstructed from the synthetic data using common image processing settings.

2) Training reconstruction models on synthetic CFA data, and comparing to RGB images demosaiced with different kinds of processing, including only demosaicing.

These approaches follow the pattern of adding noise to data, applying a CFA filter, demosaicing, and training model from the CFA and RGB data seperately (see figure 2.1).

Our study begins with the background; an overview of the principles of digital photography, demosaicing and related image processing techniques, followed by a detailed description of different noise models, and their implementations. We then describe our methodology for generating synthetic CFA data, introducing noise, implementing a denoising and signal reconstruction model and training the models. In the experiments section; we design experiments to compare the mean signal reconstruction error (MSRE) across different noise levels and types, providing insights into the effectiveness of each demosaicing method in regards to producing data for ML. We further evaluate the signal reconstruction models with the demosaicing algorithms as a baseline.

In conclusion, signal reconstruction performance for the proposed deep learning U-Net can be limited by image processing techniques such as median filtering, white balance and white level, while training on CFA often produces models that evaluate to a lower MSRE.

3. *Background*

This chapter provides an overview of the principles of a generic CFA coupled digital photography setup, four demosaicing algorithms, and image processing techniques for white level processing, white balance processing and median filtering. This is followed by a detailed description of different noise models, specifically Gaussian, speckle and salt-and-pepper noise, including consideration for implementing these models. Lastly the chapter includes on overview of current relevant literature for CFA deep learning, including data augmentation, data synthesis, and data interpretation (ei. encoding).

3.1 Digital Photography

In most modern cameras, light is captured by one array of light sensors, photosites, that convert light intensity to a digital output. Since light intensity doesn't capture which distribution of wavelengths the photosite is exposed to, a color filter array (CFA), also called a mosaic, is placed so it intercepts the light before it reaches the sensor. A CFA enables a photosite to measure the intensity of a band of wavelengths. A CFA can be arranged in what is called a Bayer pattern; a repeating pattern, 2 tiles wide, 2 tiles tall, of green, blue, red, and green filters.

Figure 3.1: The Bayer CFA pattern

The Bayer CFA enables a sensor to sample red, green and blue colour bands in different places. Reconstructing a complete signal from these samples, is called demosaicing.

3.1.1 The data-processing inequality

For many data-processing systems, like digital photography, the processing path, can be described as a Markov-chain. Defining the first variable of the chain (X_{radiance}) to be the scene radiance, the light in the scene, the next (X_{sensor}) the digital values read from the analogue-to-digital converters (ADC) on the sensor chip, and the demosaiced image as the last variable (X_{image}). This model of a digital photography system forms a chain where the next step only depends on the current step, thus fulfilling the Markov property, and forming a Markov chain;

$$X_{\text{radiance}} \rightarrow X_{\text{sensor}} \rightarrow X_{\text{image}} \tag{3.1}$$

Modelling the process this way, allows for reasoning about the informational content at each step. Here information is a measure of entropy. Capturing multiple colour bands at a single photosite with a CFA coupled sensor is not possible. As a result, the camera only captures a subset of the colour information. To reconstruct a full-colour RGB image, a demosaicing algorithm is used. The reconstruction problem is to produce an image variable as close as possible to the radiance variable, based on the sensor variable.

Knowledge of the sample (X_{sensor}) reduces the uncertainty of the radiance variable (X_{radiance}), since they are not independent. This is written as $I(X_{\text{sensor}}, X_{\text{radiance}})$ and the quantity I is called mutual information. Given that the mutual information between light and RGB image is $I(X_{\text{radiance}}, X_{\text{image}})$, the variable ordering from 3.1 and the data-processing inequality[2, p.34], it follows that the mutual information between the radiance and the sensor reading is greater than or equal to the mutual information between the radiance and the image (see equation 3.2).

$$I(X_{\text{radiance}}, X_{\text{sensor}}) \geq I(X_{\text{radiance}}, X_{\text{image}}) \tag{3.2}$$

The steps to produce a displayable image from a sensor reading cannot increase the informational content of the original sensor reading.

3.1.2 Energy and Information

If the image does not look like what was photographed, the processing is disqualified for use in producing an image. In other terms, informational content (ei. entropy) is irrelevant to a viewer, if the image does not resemble the scene radiance. Optimizing for human viewing, may be done at the cost of information.

Later, experiments will be introduced, that use mean squared reconstruction error (MSRE) as the primary metric of interest. Choosing this metric above information theoretic quantities such as mutual information (MI), yields a practical advantage, namely that calculating the second order statistic, MSRE, is far more feasible than calculating the higher order MI.

The rationale for using MSRE is, that a reconstruction process must effectively utilize information to ensure accuracy across a broad spectrum of inputs. In this way, a MSRE based experiment gives insight into the processing chain, without relying on exotic estimates of MI.

3.2 Noise

Adding noise to images and processing them, can show how algorithms perform under non-ideal conditions. This noise should therefore model and amplify, the noise expected in real-world non-ideal situations.

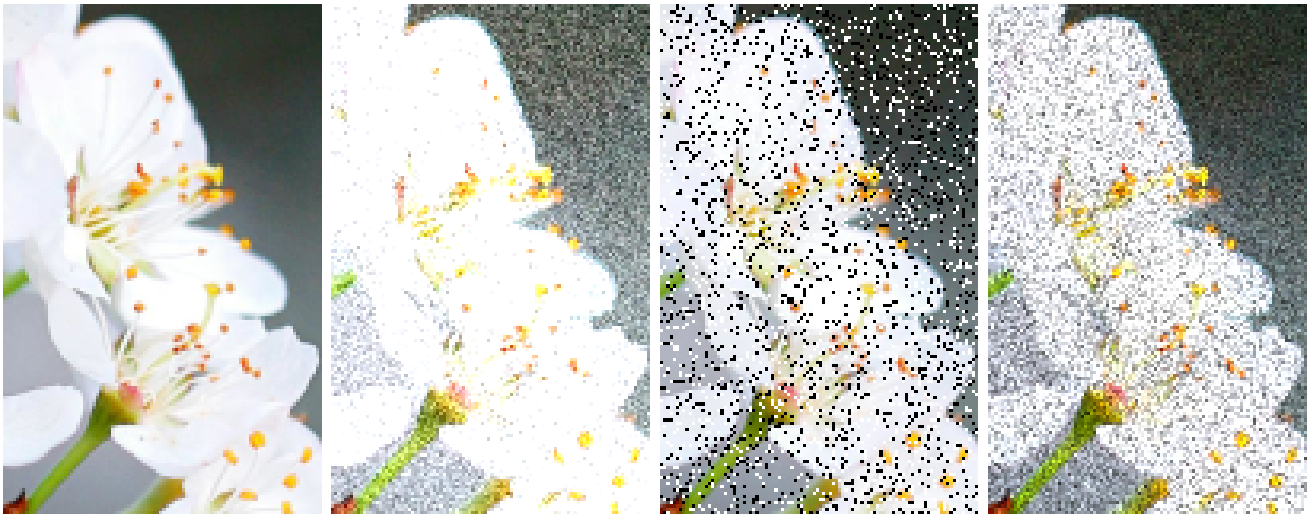

Figure 3.2: Visual comparison between different noise types. From left to right; original image before adding noise, Gaussian, salt-and-pepper, and speckle noise. Picture by Hiếu Hoàng.

3.2.1 Gaussian Noise

Additive Gaussian noise is often used to model thermal noise present in electrical circuits, and worst-case photon-counting-noise[3]. This noise type is modelled as the addition of a normally distributed noise term (n) atop the underlying image signal (p) of every pixel (x);

$$f(x) = p(x) + n(x) \tag{3.3}$$

Since the noise term (n) is defined with a non-zero probability for all outcomes, it is possible for the resulting signal to be negative. In these situations the signal can be truncated to keep $f \geq 0$[3].

Implementing this noise type should be done by drawing one random value per pixel from a Gaussian distribution, adding the original signal and noise at a higher bit-depth than the input signal, to avoid integer overflows.

3.2.2 Salt-and-Pepper Noise

Salt-and-pepper noise is used to model digital transmission noise, where individual bits may be lost[3]. Individual pixels take on either the maximum or minimum possible value allowed by its encoding, eg. 0 or 255 for 8-bit images. The noisy signal is distributed, as described by the following equations, where a is the probability of a pixel being noisy, p is the input signal, p_{MIN} is the minimum possible value of p, p_{MAX} the maximum value, and f the resulting signal [3].

$$P(f = p) = 1 - a \tag{3.4}$$

$$P(f = p_{\text{MIN}}) = \frac{a}{2} \tag{3.5}$$

$$P(f = p_{\text{MAX}}) = \frac{a}{2} \tag{3.6}$$

To implement salt-and-pepper noise, a unique set of random coordinates are drawn. Half are used to set pixels to p_{MIN} and the other half to set pixel to p_{MAX}.

3.2.3 Speckle Noise

Speckle noise inherently exists in images produced by coherent imaging systems such as laser, radar, and ultrasound. Speckle noise is characterized by a granular 'speckle' pattern, that is dependent on reflections in the imaged surface.

The mathematical by-the-book model for an image signal corrupted by speckle noise is as follows[3]:

$$f(x) = p(x) \cdot n(x) \tag{3.7}$$

Where n is exponentially distributed, with mean 1.

Another model, which is often implemented as speckle noise, is:

$$f(x) = p(x) + p(x) \cdot n(x) \tag{3.8}$$

The latter model is how speckle noise is implemented in `MATLAB`, where n is uniformly distributed[4] and in `Scikitlearn` where n is normally distributed[5]. The advantages of adding noise this way is that the variance can be controlled without shifting the mean, allowing the noise intensity to be adjusted. The study follows the `Scikitlearn` model.

3.3 Demosaicing

Demosaicing is a term that refers to methods for reconstructing and estimating an RGB image from a CFA capture. Although this section is limited to signal reconstruction, it is important to note that novel techniques for creating RGB images, that do not rely on a CFA, exist. For example pixel shift cameras, film scans and photography using pan-sharpening. These methods of capturing deviate from the digital photography model we use, which is based on a CFA sensor. These other methods are therefore out of scope for the study.

Furthermore we limit this section to demosaicing the Bayer pattern.This excludes other CFA layouts, like Fujifilms X-trans, or the cyan-yellow-green-magenta (CYGM) sensor used in some Nikon and Canon cameras, among other CFA glayouts.

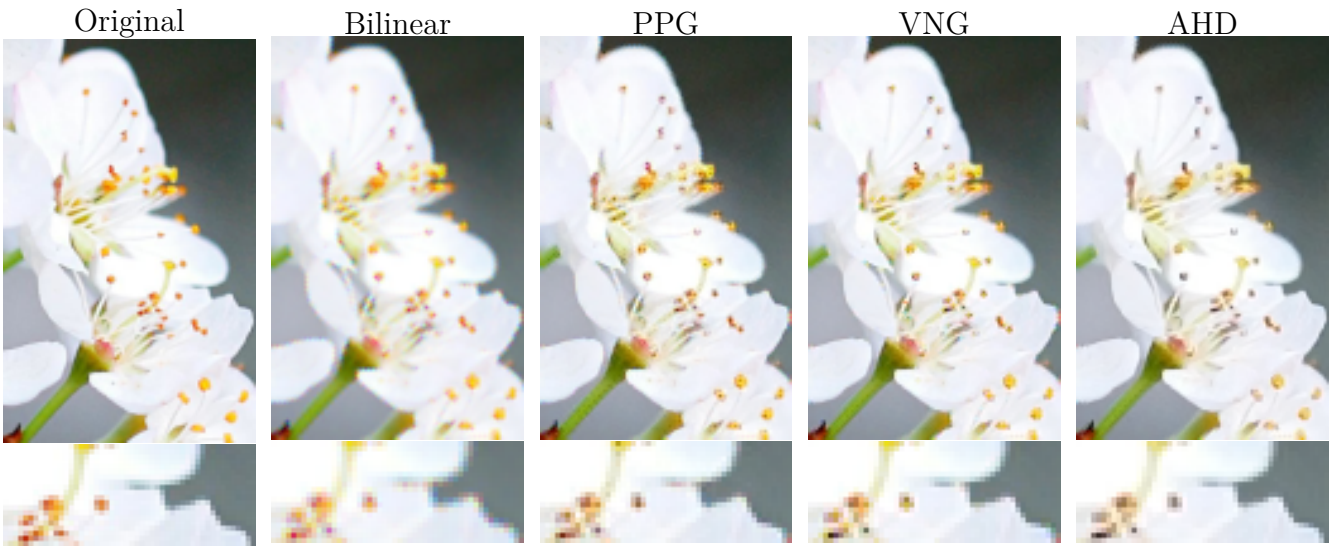

Figure 3.3: Visual comparison between different demosaicing algorithms. Picture by Hiếu Hoàng. Present in the images are false colors around the edges of the petals in the bilinear picture, and a reduction in saturation of the stamen on the AHD picture, when compared to the original image.

3.3.1 Bilinear demosaicing

A basic demosaicing algorithm is the bilinear interpolator. The signal is reconstructed by fitting a line between two values and calculating the new signal values using this line for a 2d signal. This can be done in one direction then the other, thereby bilinear. Bilinear interpolation is very close to blurring each channel separately. Like a blur, the bilinear interpolation for the Bayer pattern CFA can be done using convolutions with the following kernels:

$$K_{red} = K_{blue} = \begin{bmatrix} 0.25 & 0.5 & 0.25 \\ 0.5 & 1 & 0.5 \\ 0.25 & 0.5 & 0.25 \end{bmatrix}, K_{green} = \begin{bmatrix} 0 & 0.25 & 0 \\ 0.25 & 1 & 0.25 \\ 0 & 0.25 & 0 \end{bmatrix} \tag{3.9}$$

This blur is visible, as the interpolation smooths out detail (see figure 3.3). Since the interpolation is done independently for each layer, the bilinear "blurring" produces false color artifacts along edges. This is visible as orange and blue color at the edge of the petal in figure 3.3.

3.3.2 Patterned Pixel Grouping (PPG) demosaicing

Patterned Pixel Grouping (PPG), also referred to as Pixel Grouping, is a demosaicing algorithm that utilizes a small number of gradients to guide demosaicing. Introduced by Lin[6], the algorithm works by interpolating the missing green pixels first, calculating four gradients and interpolating in the direction with the smallest one.

Interpolation of red and blue pixels, is done using a `hue_transit` function. This determines whether the pixels at a given position form an extrema relative to the interpolation direction, and interpolates red and blue while preserving the extrema.

3.4 Variable Number of Gradients (VNG) demosaicing

The Variable Number of Gradients (VNG) algorithm was introduced by Chang, Cheung, and Pan[7] and works by calculating different edge gradients and combining color information around the gradient regions to estimate new color values.

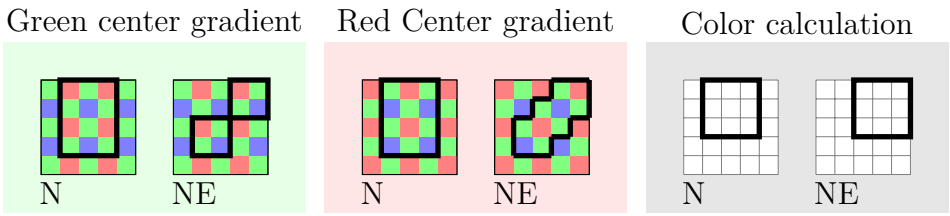

Figure 3.4: VNG gradient and color calculation regions, only North and North East regions are included. Other regions can be constructed by rotating these regions by 90, 180, and 270 degrees.

The gradients used correspond to 45 degree intervals, like that of a compass. The northern (N) gradient going from the center pixel upwards, is calculated using the differences in a 3 pixel wide and 4 pixel tall box (see figure 3.4). The gradients are calculated differently, depending on what color the CFA is at the center of the considered region (see figure 3.4).

A threshold for what gradients to include is calculated as follows:

$$T = k_1 Max + k_2(Min + Max)$$

Where Max is the maximum gradient value, Min is the minimum gradient value, k_1 and k_2 are parameters, which are set to $k_1 = 3/2$ and $k_2 = 1/2$ in the original paper [7].

For all gradients, a corresponding color calculation region exists. A region is used if its corresponding gradient is included. Sums R_{SUM}, G_{SUM} B_{SUM} are calculated based on the color in the regions. From this a color difference is calculated and used to estimate new color values.

$$r_i = g_i + \frac{(R_{SUM} - G_{SUM})}{n_{gradients}}$$

In the above equation r_i is the estimated red pixel value at position i, g_i is the measured green value at position i and $n_{gradients}$ the number of included gradients. In the way VNG functions, it is assumed that the color difference is slowly changing, since it uses this difference to compute new color values.

3.4.1 Adaptive Homogeneity-Directed (AHD) demosaicing

The Adaptive Homogeneity-Directed (AHD) algorithm is a demosaicing algorithm introduced by Hirakawa and Parks[8], that uses adaptive parameters, homogeneity-maps, and directed demosaicing to produce RGB images from CFA data.

It is assumed that the color difference between red and green $(R - G)$ and between blue and green $(B - G)$ is smaller than the difference in luminance, when measured across a small area.

The algorithm works by combining two different interpolations which are constructed in the following way:

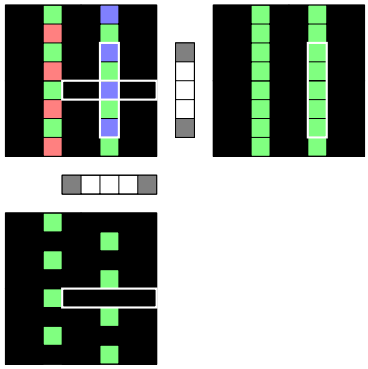

Figure 3.5: Illustration of the convolution used to interpolate green samples in AHD

1. Interpolate the green channel, using two 1 by 5 pixel kernels (see figure 3.5).
2. Calculate the color difference at red and blue pixels.
3. Interpolate color differences (e.g. $R - G$), to get one difference per pixel, using bilinear interpolation.
4. Add the interpolated green channel to the color differences, to get red and blue channels (e.g. $G + (R - G) = R$).

Combining these two RGB interpolations is done by calculating the homogeneity at each pixel for both interpolations. The most homogeneous pixels are used in the final image.

Homogeneity is calculated as the number of pixels, that are within a certain spatial distance, a certain chrominance distance, and a certain luminance distance from the considered pixel. The homogeneity map is then smoothed using a 3 by 3 pixel averaging filter. The luminance and chrominance values use L2 and L1 distance respectively, and are calculated in a perpetual colorspace, since the demosaicing is done with "*a priori* knowledge that the end user of the output image is a human"[8]. Thus, the choice of pixel is non-linear and human centered.

3.5 Processing

Each of the techniques described in the section, white balance, white level and median filter, are used to correct the image signal in ways assumed to be preferable for an human viewer.

3.5.1 White balance

White balance adjusts the image with the aim of neutralizing color differences caused by lighting conditions, such that white color appears white in the image. In CFA-based sensors, white balance can be applied in both pre- and postdemosaicing. Adjusting the white balance before the demosaicing process, balances each colour channel to the lighting conditions. This maintains colour accuracy during the subsequent interpolation process. Doing this after the demosaicing, involves adjusting the RGB values of the reconstructed image. While simpler, this can lead to less accurate color, since it works on already interpolated data, which can include inaccurate color information.

3.5.2 White Level

For digital images a limited range of values can be represented in most image file formats, limiting how much of a CFA capture can be represented in the final image. Setting a white level typically refers to setting what value from the CFA capture represents the maximum value in the processed image, truncating all values in the CFA capture above this limit.

3.5.3 Median Filter

A median filter is a non-linear digital filtering technique, often used to remove noise or color errors from an image or signal.

The median filter calculates the median for values in a pixels neighbourhood. For a 3 by 3 pixel median filter, the median is found for the eight surrounding pixels and the center pixel. The filter can be applied to individual color channels, or to other 2d planes, like the color difference between two channels.

3.5.4 Gamma Curve

A gamma curve in image processing, describes how the brightness of a image is adjusted to appear more natural for human viewing. This process, also known as gamma correction, is used to convert linear luminance captured by imaging sensors into a format that can be accurately displayed.

A typical gamma curve is represented as:

$$J = aI^{\gamma}$$

Here, J is the output intensity, I is the input intensity, a is a constant and γ is the exponent that determines the shape of the curve.

3.6 Current Literature for learning from CFA data

For our purposes, we focus on the processing steps to training, and less on the downstream tasks themselves. The section is structured into the following themes; the availability of CFA datasets, augmenting CFA datasets, synthesizing CFA data, and interpreting CFA data.

3.6.1 Augmenting CFA data

Given a CFA dataset, the CFA pattern might not be the same for every capture. In some ways of interpreting CFA data where 4 pixels are grouped (see CFA packing and BCA on page 15), this will introduce spatial errors, as the position of the color channels are shifted. This can be avoided by cropping the CFA capture such that the Bayer pattern is changed, eg. starting from blue instead of green (see figure 3.6).

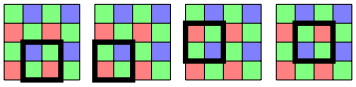

Figure 3.6: CFA augmentation cropping a GRBG sensor to other layouts.

This technique, introduced by Liu, Wu, Wang, *et al.*[9], allows datasets to be "unified" such that every capture has the same bayer pattern, at the cost of cropping each image by up to 1 pixel on each axis.

3.6.2 Synthesizing CFA data

It is possible to generate CFA data to enable experiments, without using captured CFA data directly. This means that the original dataset can be used as a baseline for signal reconstruction experiments.

Sampling RGB

The simplest way to generate data in a CFA pattern, is to sample full RGB images. This removes 2/3 of the color data, creating an image that has the mosaic structure of a CFA capture, but not any other characteristics of a non-linearized CFA signal. This approach is used in studies of demosaicing algorithms, for instance Malvar, He, and Cutler[10].

Inverse Image Signal Processing (ISP)

To model the other signal characteristics beyond the mosaic, one can estimate the original signal, by estimating inverse functions of an ISP pipeline for converting CFA captures to RGB images. This approach is used by Brooks, Mildenhall, Xue, *et al.*[11] to generate a CFA dataset for a raw-to-raw denoiser. Each step is a specific signal processing function like scaling the input by a factor. The parameters for these transformations are determined by manually examining the Darmstadt Noise Dataset[11]. Their pipeline estimates inverse functions for bilinear demosaicing, exposure correction (read "digital gain"), white balance (per channel gain), color correction, gamma correction, and tone mapping, to synthesize CFA data from RGB images.

CycleGAN inverse ISP

Expanding the inverse ISP method, Li, Lu, Zhang, *et al.*[12] introduces the unpaired CycleR2R model for simulating CFA data from existing RGB datasets. In CycleR2R each step of a ISP pipeline is estimated with CycleGAN models, a picture-to-picture translation technique, rather than preset signal processing steps.

3.6.3 Interpreting and processing CFA data

Unlike an RGB image, the shape that should be given to a CFA reading isn't predetermined. Using the CFA reading as a 2d matrix leaves out where the color filters are located. Different techniques encode this information.

CFA labeling

Given information about the CFA layout, literature about demosaicing interpret the CFA as a 2d signal using the information about the CFA mosaic to guide the demosaicing process. One way to embed this information, is to expand the signal into three separate channels, each containing all the samples of one color, with zeros in positions where the color was not sampled. This technique is implied by its use, but not explicitly established in literature. It is often done in visualizations

and figures, Hirakawa and Parks[8] use it to visualize sensor filters, and Malvar, He, and Cutler[10] use it to visualize their linear demosaicing design.

CFA packing

CFA packing rearranges a sensor reading into four 2d planes, each containing the information of a single color filter in the CFA pattern, at the cost of some spatial information (see figure 3.7).

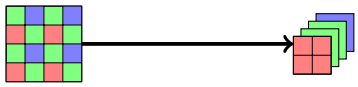

Figure 3.7: CFA packing

Many use this approach, although naming differs. Li, Lu, Zhang, *et al.*[12] use it for their raw image compression task, here called rearrangement and stacking. Brooks, Mildenhall, Xue, *et al.*[11] interpret the CFA as four color "planes". Liu, Wu, Wang, *et al.*[9], use the term "pack" directly for their work on Bayer pattern augmentation and denoising. Packing is also used in Kantas, Antoniussen, Andersen, *et al.*[13] for a classification task, and by Liang, Chen, Liu, *et al.*[14] for CFA deblurring.

Bidirectional Cross Attention (BCA) transformer

Liang, Chen, Liu, *et al.*[14] introduce a ML model for deblurring CFA data, that uses a BCA transformer to combine the packed CFA, called the color branch, with the original CFA signal, called the spatial branch. The result is then used for model tasks. This technique is also used by Kantas, Antoniussen, Andersen, *et al.*[13] as input for a classification model.

4. *Method*

In this chapter our methodology is described, the chapter is comprised of; a description of the dataset, generating synthetic CFA data from these datasets using a Bayer pattern, introducing noise, choosing demosaicing software, demosaicing software features, and implementing a denoising and signal reconstruction U-Net model. The implementation code is available on `github`[15].

4.1 CFA data

When working with CFA images, many tend to use synthetic CFA data instead of working with real world data. This is also our approach. Synthetic data provides a controlled environment, where noise can be precisely defined. Furthermore, the original data can be used as the target for signal reconstruction tasks. Varying capture conditions and sensor differences also make consistent evaluation difficult on real world data.

Lastly, creating synthetic CFA data allows for greater data availability and diversity. This allows us to generate large-scale datasets under a wide range of noisy conditions.

Figure 4.1: Subset of 25 center-cropped images from each dataset scaled to 50 by 50 pixel per image

The datasets used (see figure 4.1) and the names used to refer to them are:

- `Imagenette`: Refers to a small subset of 10 categories of the larger `ImageNet`, containing pictures of various objects. The specific version used, one down-scaled to 160 pixel on the smallest axis. The dataset is chosen because of the small image size, which provide steeper edge gradients compared to larger image sizes. The dataset has a canonical train and validation split of 9 469 and 3 925 images respectively.
- `CelebA`: Refers to a down-scaled version of the `CelebA-HQ` dataset, which contains picture of celebrity faces. The images are 256 pixel wide and 256 pixels tall. This dataset is

chosen because, the human face is an important optimization target for imaging systems. The training and validation split used is 20 000 training examples and 10 000 validation examples.

- `EuroSAT`: A dataset of cropped satellite images containing different land cover categories. The dataset was chosen since it contains photographs that are not meant for human viewing only. The training and validation split used, is the categories `Residential`, `River`, and `SeaLake` for validation, and all other categories for training.

4.2 Creating "raw" digital negative files

To be able to use common implementations of demosaicing algorithms, the data has to be loaded into a portable container file. Most "raw" formats are proprietary and specific to each camera vendor. The digital negative format (DNG) is standardized by Adobe [16] and can be implemented by anyone. It is therefore a good target format for our purposes, since available software can be used to process these files.

To write a `.dng`, a python library called `PiDNG` is used, while `RawTherapee` is used to test whether the resulting file is written correctly.

The combination of `PiDNG` and `RawTherapee` has raised some problems, leading to workarounds of software bugs; although `PiDNG` supports compression of `.dng` files, enabling this feature means `RawTherapee` cannot open the file. Also, `RawTherapee` seem to be unable to open files smaller than 225 by 225 pixels, the dimensions were found using binary search and only square images were tested (see section 4.3.1).

4.2.1 Synthetic data and colour-profiles.

The method chosen for generating synthetic CFA data is sampling RGB images, since this method allows noise to be added before and after the sampling process. Other methods model other properties of a CFA capture other than the mosaic, meaning that adding noise before, is different than adding noise after creating the mosaic, this would limit the generation of baseline models to compare model performance to. Without a baseline, calibrating the training is difficult since no insight is given to how well the models can theoretically perform.

Generating synthetic demosaic-ready Bayer like sensor data, is done by making a 3x4x2x2 kernel that maps one pixel from a color channel to a new layer. For RG-GB, this results in four layers, including a distinct layer for each green channel. A transpose convolution of the same kernel is then applied, yielding an RGB image containing 2/3 of the original color data (see figure 4.2).

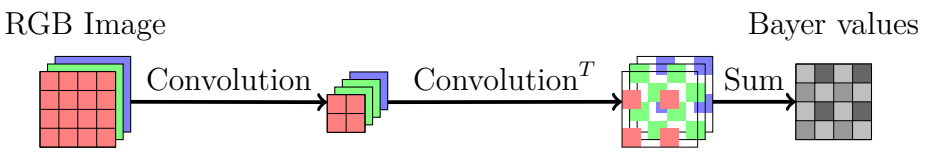

Figure 4.2: Synthetic Bayer pipeline

Interpreting data this way generates a 3d array that can be summed into a 2d Bayer like image. This approach does not account for color-profile information that might be in the file. Since the

datasets used does not come with identifiable ICC color profiles, that is non where found when running `exiftool -ICC_Profile`, and JPEG does not specify a standard profile[17], it is assumed that the ICC color-profile is `sRGB`.

4.3 Augmenting datasets

To enable our experiments it is necessary to have CFA like data, add noise, and demosaic the noisy CFA data to RGB images. This CFA data can be simulated from RGB images. The noise types added are Gaussian, speckle and salt-and-pepper noise, since they model common noise types digital imaging.

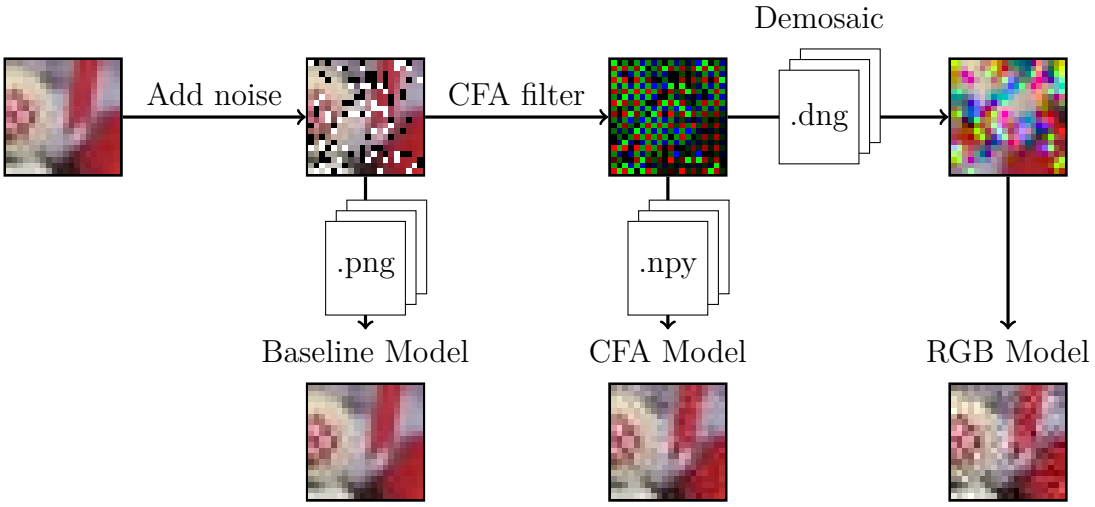

Figure 4.3: Experiment pipeline

When determining the optimal place to add noise, the choice is between adding noise in the input signal or the measurement process, ie. read noise, before or after applying the CFA process. However, since our CFA process only samples the input signal, and the noise has the same magnitude on each channel, adding noise before or after applying the CFA filter yields the same result. Therefore noise is added before the CFA process allowing baseline model to be trained on noisy data.

Noise generation is implemented using `numpy`, in accordance with the formulas in the previous noise section (see page 8). The noisy signal is calculated in a high bitdepth to avoid overflow and the result is truncated to the original bitdepth afterwards. A `python` script produces center-cropped images and adds noise, then saves the result as a `.png` file for training baseline models, a `.npy` file with a CFA filter applied for training CFA based models, and a `.dng` file with a CFA filter applied for demosaicing, the resulting image is then used to train models based on this demosaiced data (see figure 4.3).

4.3.1 Choosing demosaicing software

Many implementations for demosaicing CFA data exist. However few implementations make it feasible to process substantial datasets. The choice of software is therefore limited to programs, that can be automated using a scripting language.

	AHD	AMaZE	Bi-linear	EA	LMMSE	PPG	VNG
RawTherapee	yes	yes	-	-	yes	-	yes[†]
DCRaw	yes	-	yes	-	-	yes	yes[†]
Darktable	-	yes	-	-	yes	yes	yes[†]
OpenCV	-	-	yes	yes[‡]	-	-	yes

[†] `Darktable` and `RawTherapee` support only VNG4, which we interpret to be VNG with 4 color channels, where the two green pixels in the bayer sensor pattern is treated as a separate colors. `dcraw` supports both VNG and VNG4.
[‡] What algorithm, the EA, "edge-aware" demosaicing in OpenCV uses, is not referenced in the OpenCV documentation [18].

Table 4.1: Subset of supported demosaicing types per program. From left to right the algorithms are; Adaptive Homogeneity-Directed (AHD), Aliasing Minimization and Zipper Elimination (AMaZE), Bi-linear, Edge-Aware (EA), linear minimum mean square-error estimation (LMMSE), Patterned Pixel Grouping (PPG), and Variable Number of Gradients (VNG)

Among these programs, each supports different algorithms (see table 4.1). We have not been able to find descriptions of the Edge Aware (EA) and Aliasing Minimization and Zipper Elimination (AMaZE) algorithms beyond their implementation code. Some of the experiments conducted use the bi-linear demosaicing algorithm to compare linear and non-linear reconstructions. This has been a limiting factor in our choice of software.

Also influential for choosing software has been that, `RawTherapee` has trouble with small files. This issue has been brought to the attention of the software authors, and a solution has been found, the solution has not been merged at the time of writing . The issue can be resolved by recompiling `RawTherapee` with lines 9612 to 9615 of the `RawTherapee/rtengine/dcraw.cc` file removed[1].

The software we choose is `dcraw`, for it's algorithms which are all documented, it's ability to be scripted, it's processing capabilities, and it's processing speed.

`dcraw` histogram based white level algorithm

The way the white level is set by default, when using `dcraw` is using a histogram based white level algorithm.

It works by constructing a histogram of 512 bins per color [dcraw.c line 140], setting the bins to 0, and incrementing each bin by one, channel by channel, pixel by pixel [dcraw.c line 9790-9808].

Finding the white level from this histogram, is done using linear search on a running sum, adding each bin in decreasing order, light to dark. The search stops when the sum increases above `width * height * 0.01`. This finds the 99th percentile white level. This is repeated for each color channel and the smallest (darkest) value is chosen [dcraw.c line 10042-10049].

The values above the 99th percentile brightness are truncated when the new white level is applied.

This algorithm is useful for making the human eye observe images of (potentially underexposed) natural scenery better, since most of the information in the image can be displayed across a larger swath of values, when the 99th percentile brightness is the maximum value in the image file.

[1] see https://github.com/Beep6581/RawTherapee/issues/6963 for more details

`dcraw` image averaging white balance

`dcraw` offers multiple options for white balance adjustments. `dcraw` can automatically calculate white balance based on the image content. This method averages the image channels to determine the most neutral point, and adjusts the RGB channels accordingly, as per the usage instructions shipped with `dcraw` for option `"-a average the whole image for whitebalance"`.

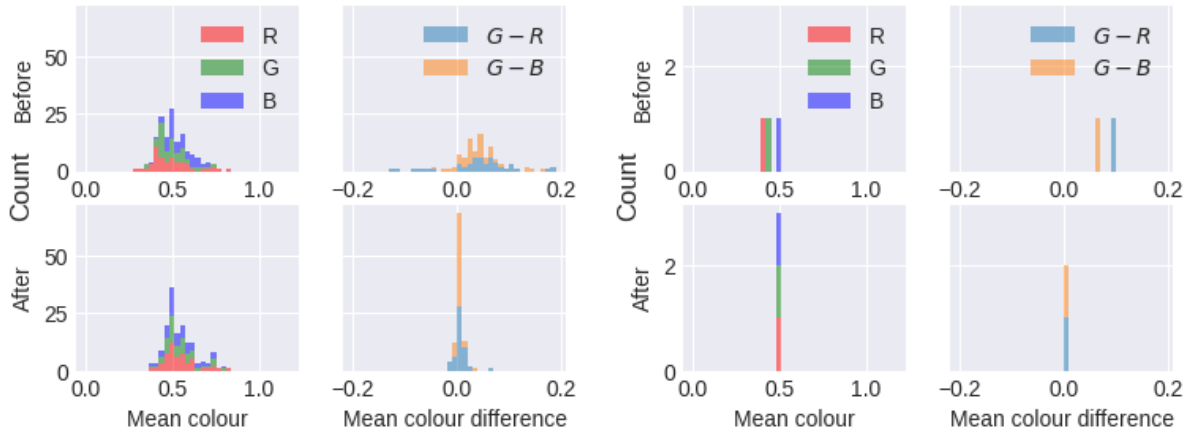

Figure 4.4: Histograms of mean colour and mean colour difference for 51 images and one image from `EuroSAT`, demosaiced with and without automatic white balance using `dcraw`

Running the algorithm on 51 images from `EuroSAT`, the algorithm scales the color channels such that the means are equal for all channels. This produces a sharp peak at 0 for mean color difference for images where the white balance is applied (see left plot in figure 4.4).

Plotting the histograms for a single image indicates that the algorithm scales up each channel, to get them to an equal level, to the brightest colour channel (see right plot in figure 4.4).

`dcraw` image median filtering

`dcraw` provides options to apply median filtering to raw images as part of its processing. The filter is a 3 by 3 median filter applied across the demosaiced color differences, as per the program usage instructions for option `"-m <num> Apply a 3x3 median filter to R-G and B-G"`.

4.3.2 Optimizing processing

Using `dcraw` in a bash script with a simple loop, the processing time per 224 by 224 pixel file is 6ms, or 0.8 mega pixels per second (MP/s), using the AHD demosaicing algorithm. For reference, `RawTherapee` performs at a rate of 0.1 MP/s for the same task. `dcraw` is single-threaded and can be parallelized using `GNU parallel` to improve the processing rate to 1.8 MP/s. The rate improves further to 2.0 MP/s when the parallel tasks are synchronized in $n_{cores} * 2$ job chunks.

4.4 Inputting CFA data into a model

Interpreting the CFA capture as a single channel will mean that the shape of the input will be different, meaning two different models are needed, one for CFA data and one for RGB data. CFA

packing presents the same problem, but additionally the height and width is halved, while a BCA transformer doubles the input layers when combining its spatial and colour branches.

For the proposed experiment CFA labeling is used, expanding the signal to 3 channels, and placing samples of one colour in each channel, with zeros where no sample was taken for that colour channel. This makes the shape for CFA data equal to that of the RGB images, enabling using the same topology for both RGB and CFA models.

4.5 U-Net

The U-net is an encoder-decoder structure, where skip connections create links between the encoder-decoder layers (see figure 4.5). The skip connections provide spatial information to the decoding, that would otherwise be lost during upscaling.

The model consists of multiple convolution layers in both the encoder and decoder parts. The encoder compresses the input image to a lower dimensional latent representation, and the decoder reconstructs images from the latent representation using transposed convolutions. The skip connections create direct transfer of feature maps from the encoder to the decoder, preserving the high resolution spatial information.

4.5.1 Denosing U-Net

In this section we describe the implementation of the U-Net structure used throughout the experiments. The architecture of the model follows Couturier, Perrot, and Salomon[1], but with 5 layers instead of 8 and L2 loss as the optimization target instead of Structural SIMilarity (SSIM).

The encoder has 5 convolution blocks, consisting of a convolution layer, a batch normal layer and a ReLU activation function.

The convolutional layer, creates feature maps, that capture different aspects of the input. The batch normalization layers normalize the feature maps, improving stability of the network by reducing the covariate shift. The ReLU activation introduces non-linearity into the model. The remaining hyperparameters, such as padding, is set so the border edge pixels are not lost in the convolution, since without the padding they would only feature in a single receptive field. The stride is set to 2 for every layer. These blocks and layers are designed as seen in the left part of figure 4.5.

The decoder restores the spatial dimension while reducing the depth of the feature maps from the encoder. It consists of 5 transposed convolution blocks, which each have a transposed convolution, batch normalization, and ReLU activation. The bottom also has a dropout layer to reduce overfitting by randomly setting a fraction of the inputs to 0 at each update during training. This prevents the network from becoming to dependent on a particular set of neurons. The output layer has TanH activation, clamping the output between 0 and 1. These blocks and layers are designed as seen in right part of figure 4.5.

The network requires, that the input is preprocessed. Assume the input image has size $N \cdot N$, then after 1 downsampling layer, we have $\frac{N}{2} \cdot \frac{N}{2}$, downscaling the features by a factor of 2. Since the depth of the U-Net is 5, we end with $\frac{N}{2^5} \cdot \frac{N}{2^5}$. For the network to downsample correctly and reach the lowest layer, we need the N to be divisible by $2^5 = 32$, since the amount of features at each encoder and decoder layer need to be equal for their skip connection to work.

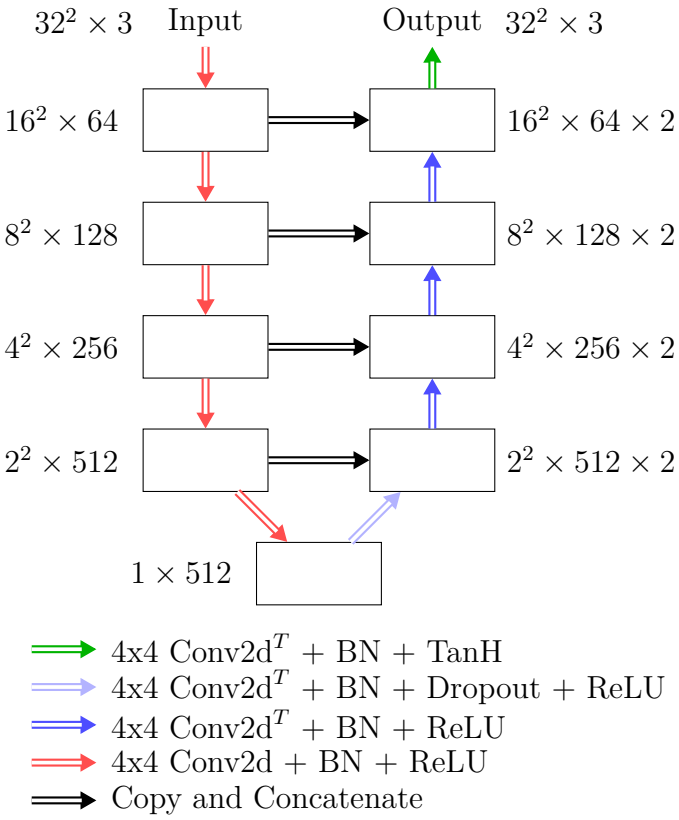

Figure 4.5: Diagram of the U-net architecture

Training the models

The network is implemented using PyTorch. The network is trained using the `AdamW` optimizer with a learning rate of 1^{-4}, and optimizes for Mean Squared Error (MSE) loss. The training loop uses batch learning with batch size 32 and the dataset is shuffled once per epoch.

The models are trained using the `hendrix` compute cluster using `slurm` job management, and `Nvidia` half-precision floating point capable GPU accelerators. When training using half-precision floats, the forward pass is done in half-precision, while the backward pass is done in full-precision. A gradient scaler is also added to avoid underflow for the half-precision training.

5. *Experiments and Discussion*

In this section; experiments are presented to compare the mean signal reconstruction error (MSRE) across different noise levels and types, providing insights into the effectiveness of each demosaicing method, when it comes to processing data for use in ML. We further evaluate the signal reconstruction models to the demosaicing algorithms as baseline.

The experiments are done by applying a CFA filter to an existing RGB dataset, adding noise, and demosaicing them (see figure 4.3). Then a denoising model is trained after each processing step. That makes 3 models one RGB denoiser, one CFA, and one denoiser trained from the demosaiced data. This is done for every noise intensity. In this way, there is a model, that acts as a baseline, which the RGB and CFA models can be evaluated against.

5.1 Initial experiment

This experiment is designed to test whether it is possible to create denoising models that performs better using CFA data, compared to using demosaiced AHD data. Using common image processing settings mimics consumer camera usage, the first experiment is done with default `dcraw` parameters.

The initial experiment was done using the `Imagenette` dataset (see page 17).

The data is then further processed in the following ways;

1. **The data is center cropped** to 160 pixels on each axis.
2. **Noise is added** Gaussian, speckle and salt-and-pepper, for each noise intensity.
3. **RGB sampling is applied to generate CFA data**
4. **The CFA data is demosaiced using AHD** via `dcraw` with histogram based white levels, sRGB gamma, and 3 iterations of 3×3 median filtering.

Denoising U-Net models are then trained on the training split of `Imagenette`. This is done for each noise type and intensity. Each model is validated on 50 random draws from the validation set. Each draw consists of 4 images, which validation loss is averaged, to form a normal distribution by the central limit theorem.

Results

The histograms 5.1 indicate a tendency that AHD demosaicing yields higher loss than training on CFA data directly, while there is little difference between the CFA and baseline models. Applying the parametric student t-test to the CFA based models validation loss and the AHD based models loss, there is a significant difference (t-test $p < 0.05$) for most models except with the 10% salt-and-pepper noise and 70% speckle noise.

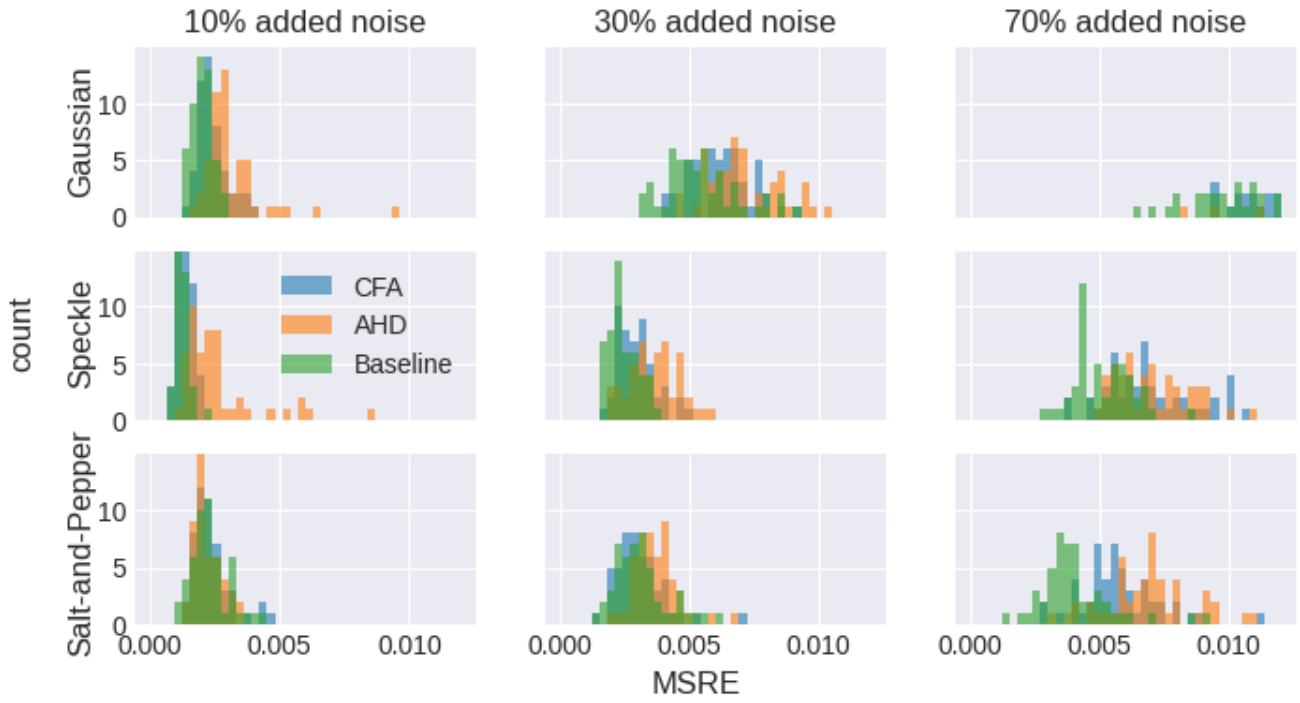

Figure 5.1: Histograms of 50 draws of MSRE across 4 samples, each histogram represents one model trained for 100 epochs

Discussion

The results indicate a trend, but the results are limited due to the number of noise intensities tested. It is unclear whether this trend exists generally.

5.2 Expanding initial Experiment

The experiment was then expanded to test noise levels from 10% to 90% in 10% intervals. This is done to test whether the trend observed in the last experiment is present at more noise level.

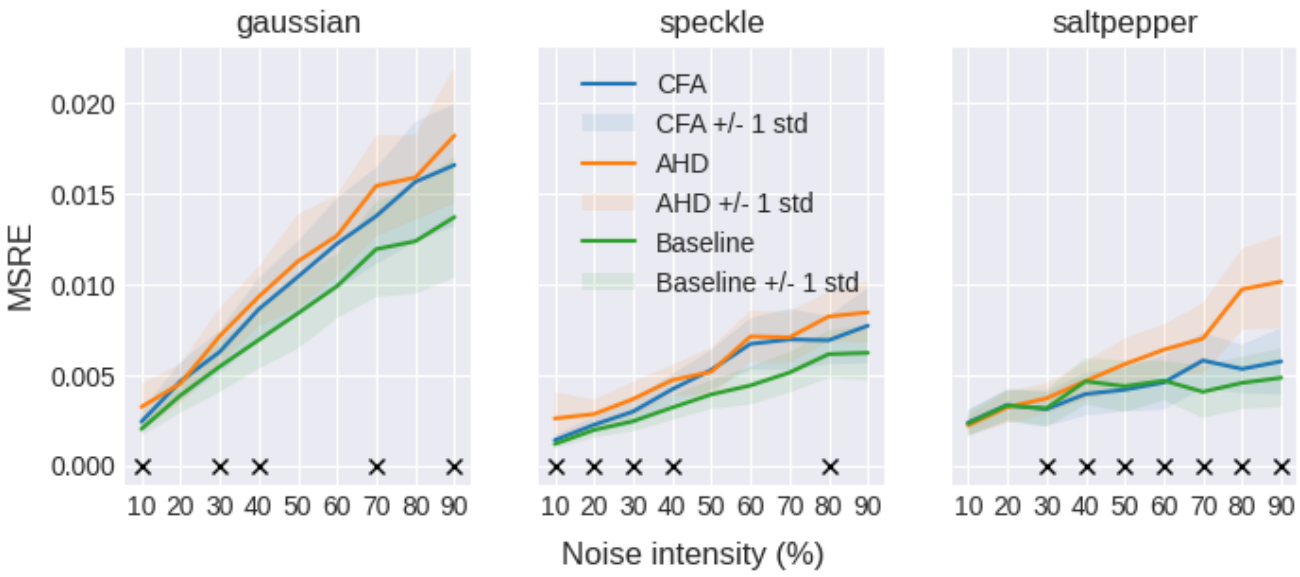

Figure 5.2: Mean and uncertainty of normal distributions fitted to 50 draws of mean squared validation error across 4 samples. Each line represents one model trained for 100 epochs. The "X" annotation denotes a significant difference (t-test $p < 0.05$) between the CFA and AHD model loss, for the noise intensity.

Results

This experiment shows that it is possible to train models that follow the same trend, across more noise intensities. The models based on CFA data perform slightly better than those based on AHD data. For salt-and-pepper this trend is only significant for high amounts of noise, while in Gaussian and speckle the significance is mixed. The gap between CFA models and the baseline models is smaller than the difference between CFA and AHD models.

Discussion

The mixed results for Gaussian and speckle makes it difficult to see any robust trends. The only small trend is in the salt-and-pepper noise model performance, where there are differences for high intensities of noise. The variance from the validation approach might be limiting the results.

5.2.1 Reconstruction Error Experiments

To address limitations of the previous experiment, a new experiment is conducted.

The previous experiments were limited in two ways. They do not address whether the training method generally produces better model performance using CFA data. It only shows that it is possible to get better performance for specific models. The other limitation is that the validation of the models is done via. random sampling. Which increases the variance of the results.

Multiple models were trained at each noise intensity to test whether the training leads to better performance. Validation is done on the entire validation set without sampling. Since there is no sampling, the assumption of normality in the loss distribution is no longer valid. Significance is therefore tested with the non-parametric Mann-Whitney U-test. This has the added advantage, that it is possible to test significance on fewer samples than the parametric Student t-test.

The data is processed in the same manner as in the first experiment. But this time the added noise intensities are 64, 32.0, 16.0, 8.0, 4.0, 2.0, 1.0, 0.5, 0.25, 0.125, 0.0625, 0.03125. Thus creating a logarithmic intensity scale rather, than a linear one.

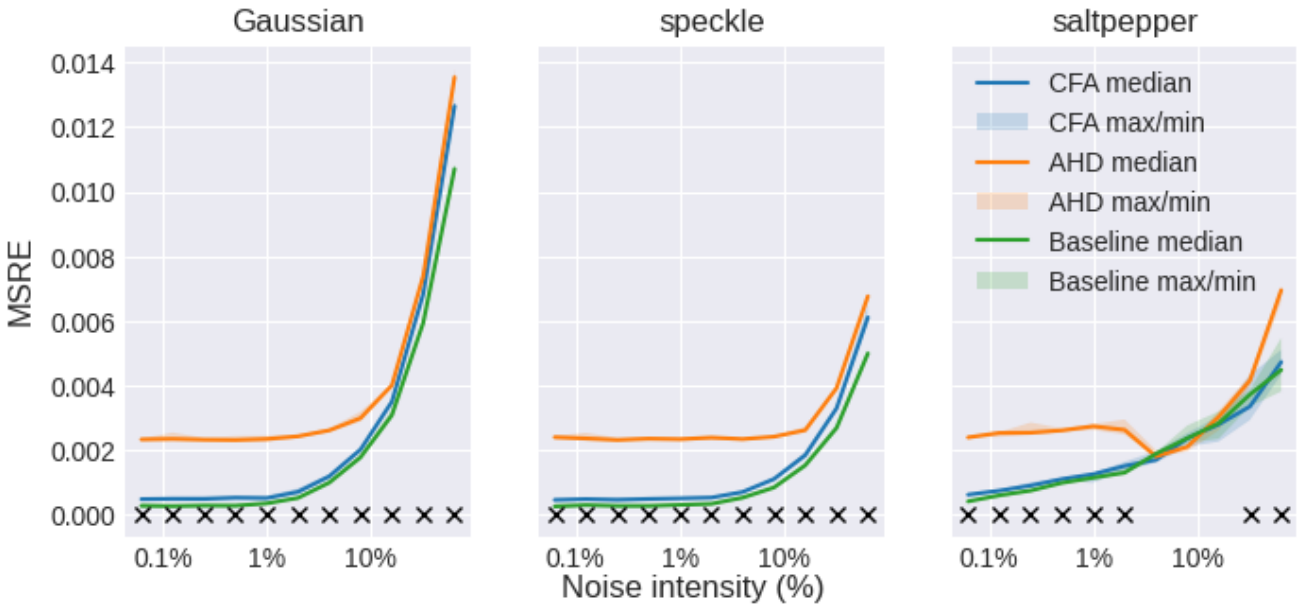

Figure 5.3: Median (line), maximum and minimum (shade) validation loss for denoising models. Every entry in the graph, correspond to the validation for 5 models, each trained for 100 epochs. The "X" annotation denotes a significant difference (u-test $p < 0.05$) between the CFA and AHD model losses.

Results

The experiments shows a gap between CFA and AHD models for all noise intensities, which is larger at lower noise intensities. With salt-and-pepper noise models between 4% and 16%, being an exception (see figure 5.4).

Discussion

MSRE for salt-and-pepper noise models between 4% and 16% is best explained by the way the `dcraw` white level algorithm works. The white level setting in `dcraw` removes information above the 99th percentile brightness. This explains the dip in loss for the salt and pepper denoising models. This is because salt-and-pepper noise adds pixels with the highest luminance encoded in the current image. The white level algorithm is covered in greater detail on page 20. Adding white pixels in the excess of 1% can make the white level histogram search end at its first iteration. Thereby setting the white level such that all information is preserved. Specifically at 4%, 8%, and 16% salt-and-pepper AHD models (see figure 5.1) the performance improve to parity with the CFA. Here the loss difference is not significant (u-test $p \not< 0.05$).

5.2.2 Reconstruction error, less processing

This experiment is designed to reduce the amount of parameters of the pipeline, reducing potential interactions between processing, to test what processing techniques decrease MSRE model performance.

The gamma curve is now set to a line, this is a better gamma curve since the input is sampled directly from the target RGB without any scaling. For a power law gamma curve ($J = aI^{\gamma}$) a line is created with following parameters $a = 1, \gamma = 1$, which is set using `-g 1 1` when calling `dcraw`.

It is recommended by Hirakawa and Parks[8] to use 3 times median filtering using a 3×3 pixel wide median filter, on the color difference channels ($R - G$ and $B - G$) when using AHD demosaicing. This is to suppress false coloration. It is possible to use the AHD algorithm without this filter.

The experiment is repeated with AHD without incorrect gamma correction, and with AHD without median filtering or the incorrect gamma curve.

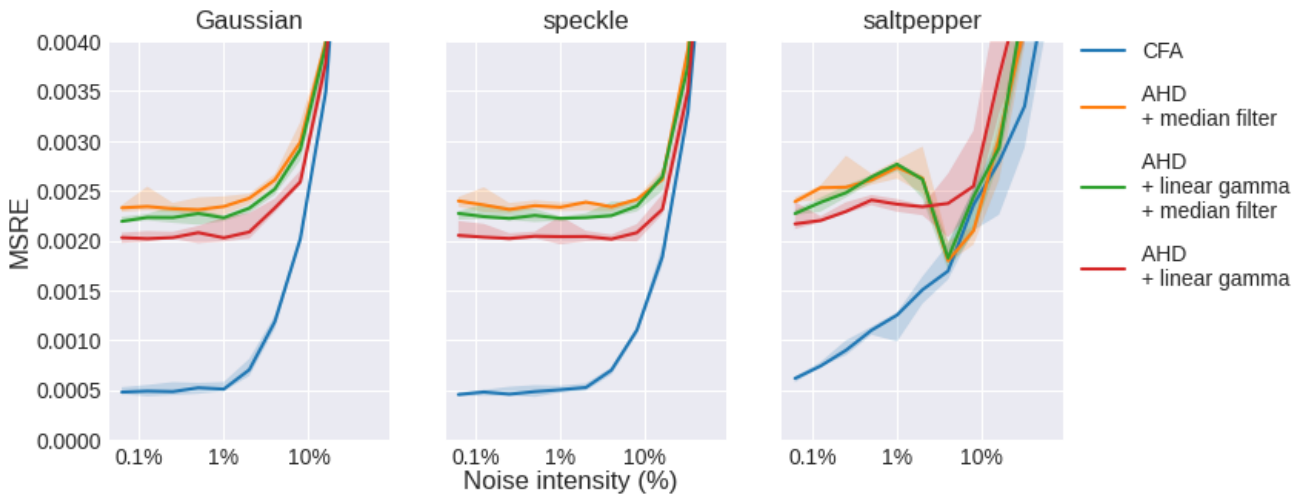

Figure 5.4: Median (line), maximum, and minimum (shade) validation loss for denoising models, each entry on each line representing five models trained for 100 epochs.

Results

Removing these two non-linear transformations reduces the loss, but does not close the gap to the CFA models. The remaining non-linear transformations done to the input is the white level setting and demosaicing.

Discussion

The dip in loss is still present around the 8% noise level with different kinds of processing. The difference between CFA and AHD models persists for other noise levels. The presence of the dip in loss around 8% indicate that the white level setting creates the loss difference rather than the demosaicing process.

5.2.3 Verifying significance

This experiment is designed to determine whether the MSRE differences comes from non-linearity in the AHD algorithm or the effect that comes from the white level algorithm.

Since the most of the differences between AHD and CFA model performance is present at low noise levels, the new experiments focus on signal reconstruction without adding noise. This simplifies the experiment setup and requires less resources, since fewer models have to be trained. In the switch to the new experimental setups, the training loop is also updated to utilize 16-bit floating point numbers for the forward pass of the model, further reducing resource use.

The following experiment is conducted by processing `Imagenette` by applying a CFA filter, then demosaicing using four different demosaicing algorithms, bilinear, PPG, VNG, and AHD. Five models are trained for each step of processing one set before processing, one set for CFA, and one set for each of four the demosaicing algorithms, yielding 6 sets of models.

Additionally the input images are cropped so the underlying CFA layout is randomized. For the CFA, the data is labelled so the information is retained. While it isn't directly available for the demosaiced data. This makes the task of determining the CFA layout more difficult, this is done to make effect of non-linear processing in the demosaicing algorithms more prominent and non-reversible. This method is the inverse of Bayer unification (see page 14). In this setup, the models cannot read the CFA directly, and have to determine what values are measured and what values are interpolated.

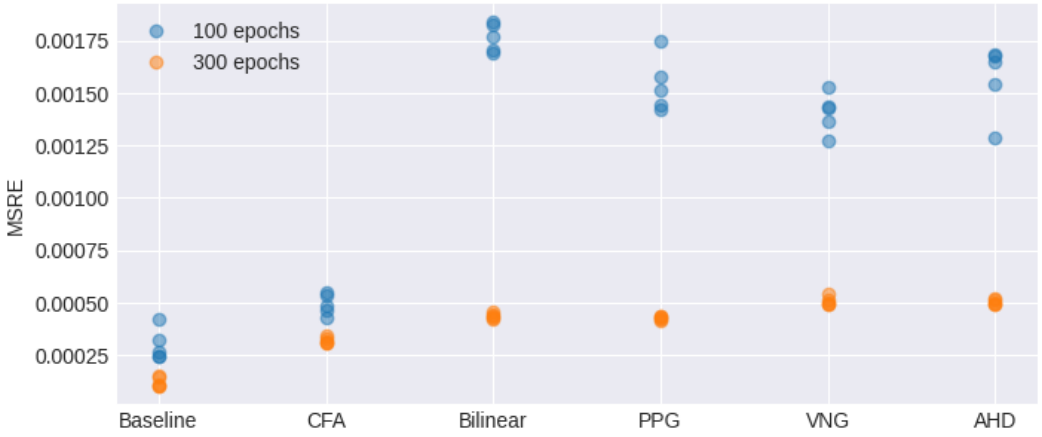

Figure 5.5: Reconstruction error with unknown CFA layout

Results

For models trained for 100 epochs, the bilinear process leads to the least accurate model performance. However this effect disappears for models which are trained for 300 epochs. Here the model performance align, so that the non-linear AHD processing steps lead to less accurate outcomes than the linear bilinear process.

Discussion

The higher MSRE observed in the previous experiment does not come from non-linear processing in the demosaicing algorithms.

5.3 Image processing experiments

Given the differences between signal reconstruction performances differ little between demosaicing algorithms, the following experiments investigate implications of different image processing steps on signal reconstruction.

5.3.1 MSRE models

To test how different image processing techniques affect U-Net MSRE, reconstruction models are now trained on processed demosaiced data. The data is processed using different combinations of white level processing, white balance processing, and median filtering. The CFA layout is randomized and multiple dataset are tested.

Imagenette

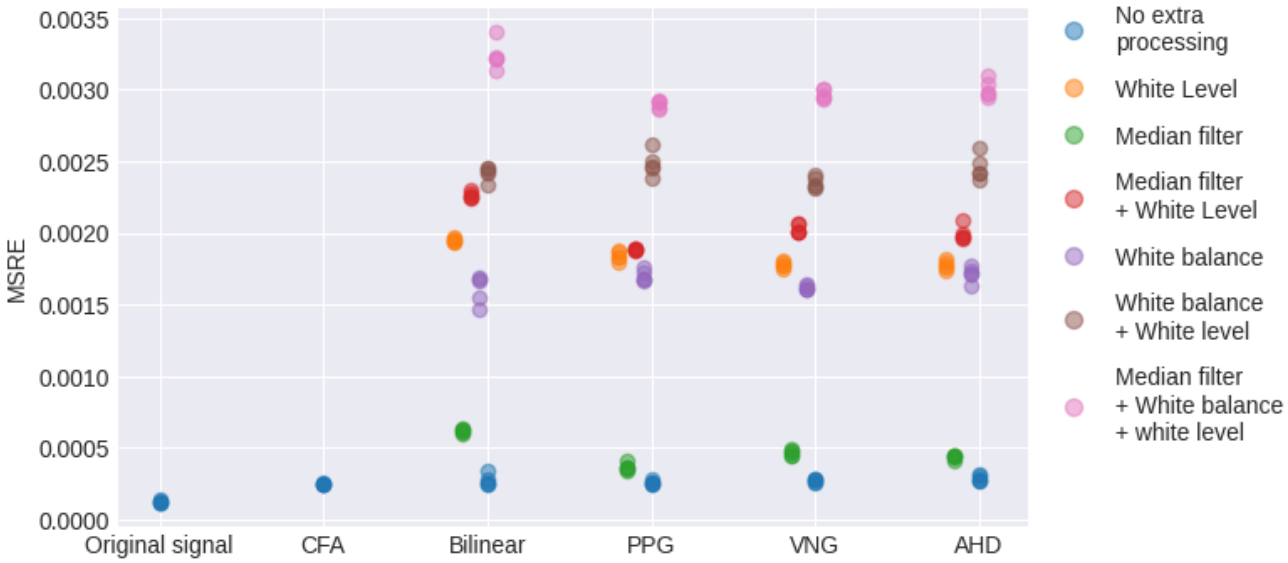

Figure 5.6: MSRE for models trained 300 epochs on `Imagenette`

When reconstructing the target signal without extra processing than demosaicing, CFA has a MSRE that is significantly lower than both VNG and AHD demosaicing (u-test $p < 0.05$), but not PPG or Bilinear demosaicing (u-test $p \not< 0.05$). All other kinds of demosaicing with extra processing yield a significantly higher MSRE than CFA.

The MSRE by processing type is similar across different demosaicing algorithms. MSRE is lowest for no processing, adding a median filter increases MSRE, white balance increases MSRE more than the median filter, and white level more than the white balance. Combining processing steps increases the MSRE further, the models that evaluate with the highest MSRE use all three types of processing (see figure 5.6).

CelebA

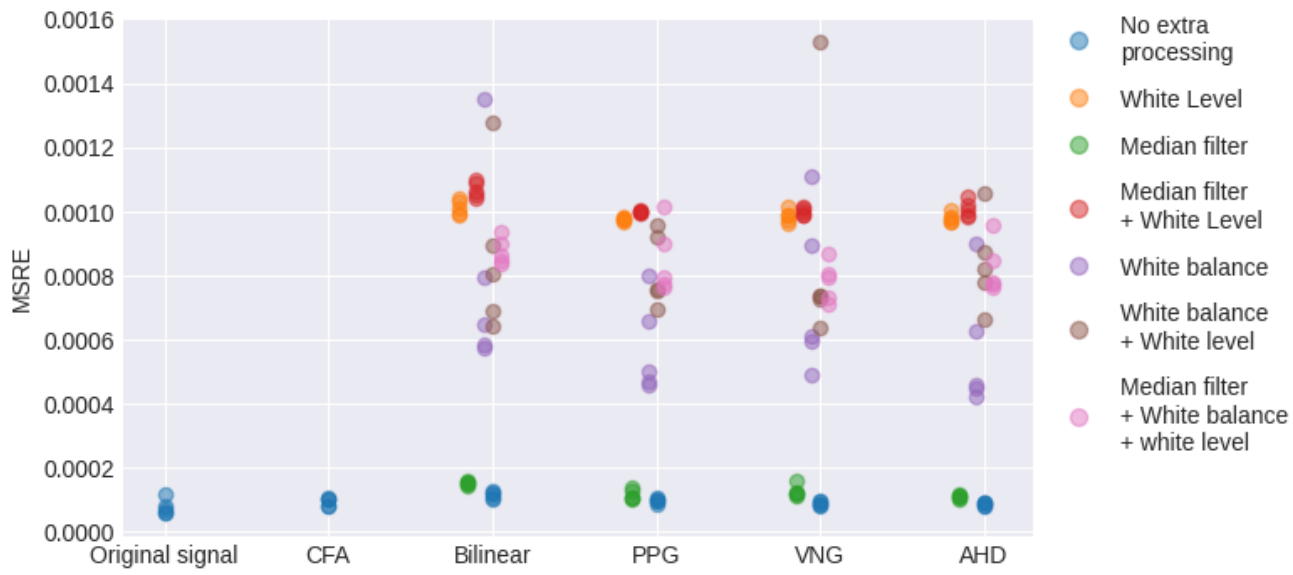

Figure 5.7: MSRE for models trained 100 epochs on `CelebA`

MSRE for `CelebA` is significantly lower for CFA than for model trained data processed with bilinear demosaicing with no extra processing (u-test $p < 0.05$). This is not present for other kinds of demosaicing without extra processing. MSRE is significantly lower between CFA models and models trained with median filtered images, that are demosaiced bilinearly and using VNG. But not for models trained using median filtered images, that are demosaiced using PPG and AHD (u-test $p \not< 0.05$). All other models trained on demosaiced data with other combinations of processing have significantly higher MSRE, than the CFA models (u-test $p < 0.05$).

The MSRE by processing type is similar across different demosaicing algorithms. MSRE is lowest for no processing. Adding a median filter increases MSRE, white balance increases MSRE even more, and white level increases MSRE the most of any single processing step. Combining processing steps does not always increase the MSRE. All of the models that are trained on images processed with white balance have a lower MSRE than the models trained on images processed with white level alone (see figure 5.7).

EuroSAT

Figure 5.8: MSRE for models trained 100 epochs on `EuroSAT`

Differences in MSRE for `EuroSat` is insignificant between CFA models and models trained on demosaiced images, without any extra processing steps (u-test $p \not< 0.05$). The MSRE difference between CFA models and models trained on median filter and the images demosaiced using PPG is also insignificant (u-test $p \not< 0.05$). All other models than the one specifically mentioned, evaluate to a significantly higher MSRE than the CFA models (u-test $p < 0.05$).

The MSRE by processing type is similar across different demosaicing algorithms. The ordering of the processing steps by MSRE, is the same as the one for `Imagenette` (see figure 5.8).

Discussion

Given that the input already has the correct channel intensities, the processing pipeline for images do not produce better training data for ML models. The information in the images are also seem to be reduced for some of these processing steps. Although the median filtering improves the MSRE between demosaiced images and the original signal, training on this data can produce models that evaluate with a higher MSRE. Given a priori knowledge that the captured CFA has a correct per channel gain distribution, the white level and white balance processing only introduces error.

5.3.2 Baseline MSRE

To establish a baseline to compare reconstruction performance, the MSRE for demosaicing algorithms is established. This MSRE is then used to interpret the MSRE of U-Net models trained on demosaiced images, potentially revealing limitations in model performance. This will provide insights into how the energy (MSRE) is distributed for the image processing steps, bringing this together with model performance can show whether lower MSRE for the input leads to better model performance.

5.3.3 Imagenette

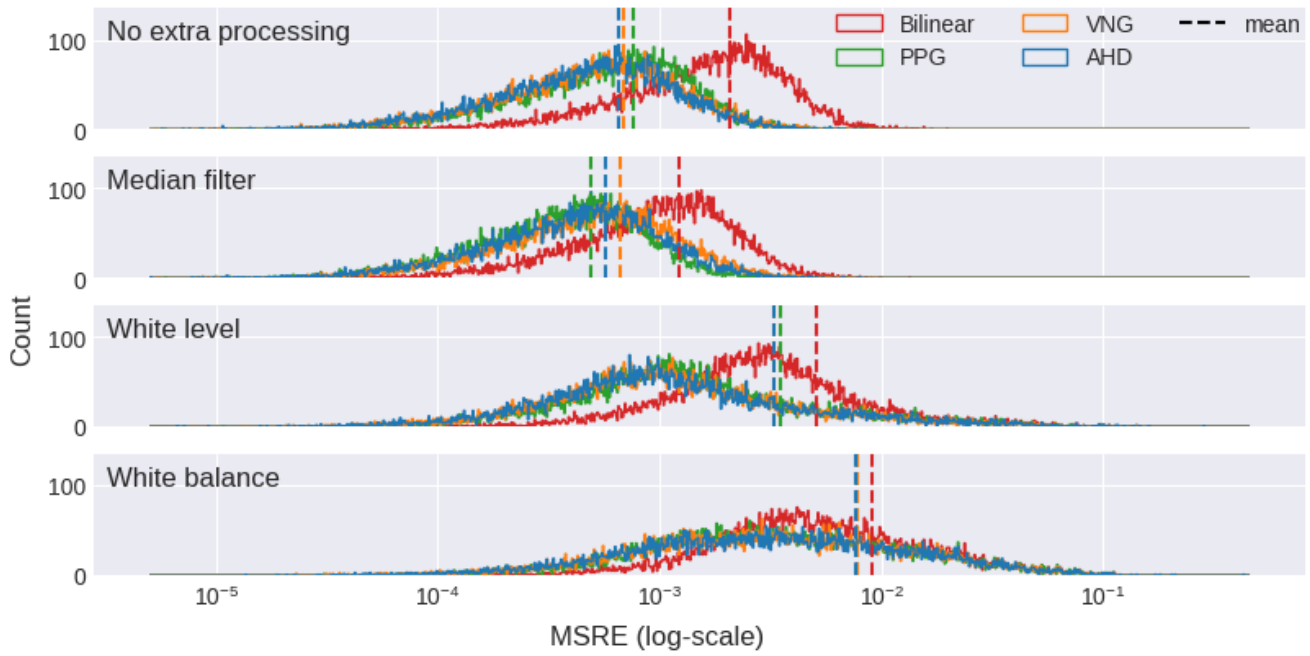

Figure 5.9: Histograms of MSRE between baseline signal and demosaiced signal for `Imagenette`

For `Imagenette` there is a trend that bilinear demosaicing produces images that resemble the original signal less, than the other demosaicing algorithms.The median filter is the only processing step that improves the energetic reconstruction, lowering the MSRE between the baseline and the demosaiced signal. The white level processing increases the MSRE and white balance increases the MSRE even further.

	CFA	Bilinear	PPG	VNG	AHD
Demosaicing	-	$3.2 \cdot 10^{-4}$	$1.0 \cdot 10^{-4}$	$8.9 \cdot 10^{-5}$	$8.0 \cdot 10^{-5}$
U-net models	$1.6 \cdot 10^{-4}$	$1.6 \cdot 10^{-4}$	$1.9 \cdot 10^{-4}$	$1.7 \cdot 10^{-4}$	$1.9 \cdot 10^{-4}$

Table 5.1: MSRE for U-Net models and demosaicing algorithms for `Imagenette`

The CFA models MSRE are higher than PPG, VNG, and AHD demosaicing for the reconstruction without extra processing, but lower than bilinear demosaicing (see table 5.1). For the models, the higher energy training data, that bilinear demosaicing provides, leads to equal model MSRE as CFA data. The lower energy (MSRE) data leads to higher MSRE when evaluating the model performance for this dataset.

CelebA

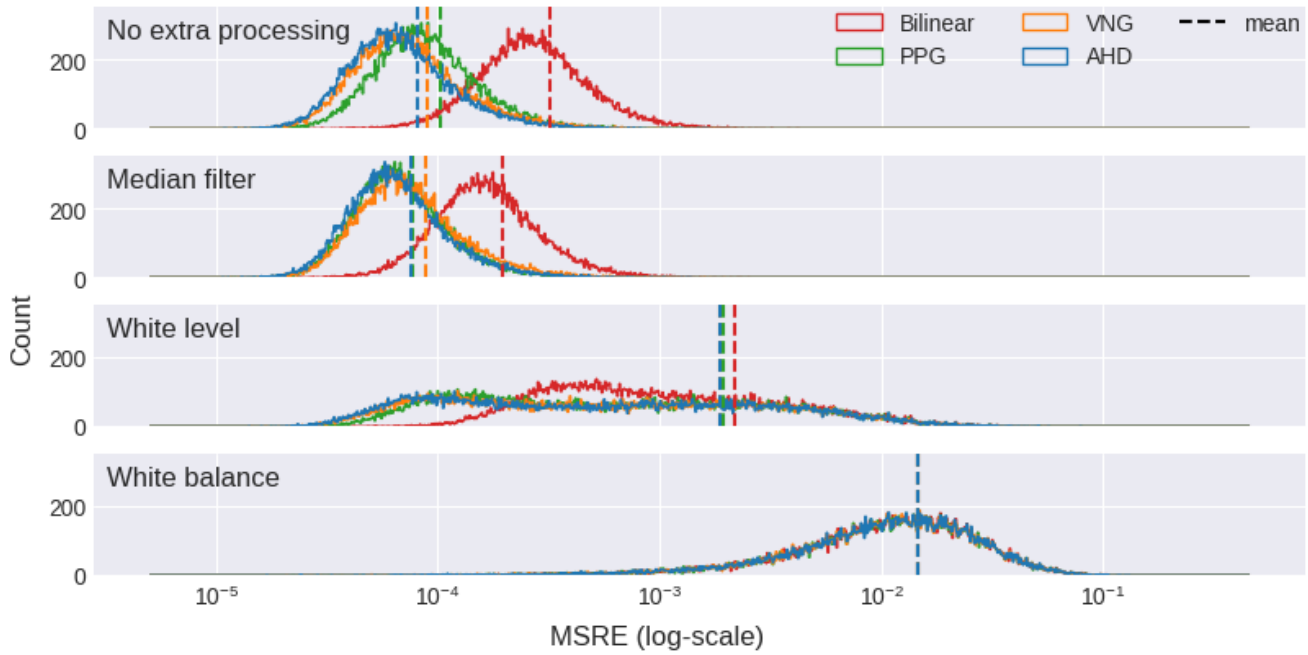

Figure 5.10: Histograms of MSRE between baseline signal and demosaiced signal for `CelebA`

For `CelebA` the trends seen in `Imagenette` also appear. The bilinear demosaicing produces images that resemble the target signal less than the other demosaicing algorithms. This effect is not present when the image is processed with white balance. Here the MSRE is equal for all demosaicing algorithms. Median filtering improves the reconstruction, lowering the MSRE compared to no extra processing. White level processing increases the loss and white balance increases the loss even more.

	CFA	Bilinear	PPG	VNG	AHD
Demosaicing	-	$3.2 \cdot 10^{-4}$	$1.0 \cdot 10^{-4}$	$8.9 \cdot 10^{-5}$	$8.0 \cdot 10^{-5}$
U-net models	$9.3 \cdot 10^{-5}$	$1.2 \cdot 10^{-4}$	$9.5 \cdot 10^{-5}$	$8.9 \cdot 10^{-5}$	$8.7 \cdot 10^{-5}$

Table 5.2: MSRE for U-Net models and demosaicing algorithms for `CelebA`

The CFA models MSRE is higher than AHD demosaicing for the reconstruction without extra processing, but equal to VNG, and lower than bilinear and PPG demosaicing (see figure 5.2). For the models, the higher energy demosaiced data, except bilinear demosaicing, leads to lower model MSRE, but on par with CFA data.

EuroSat

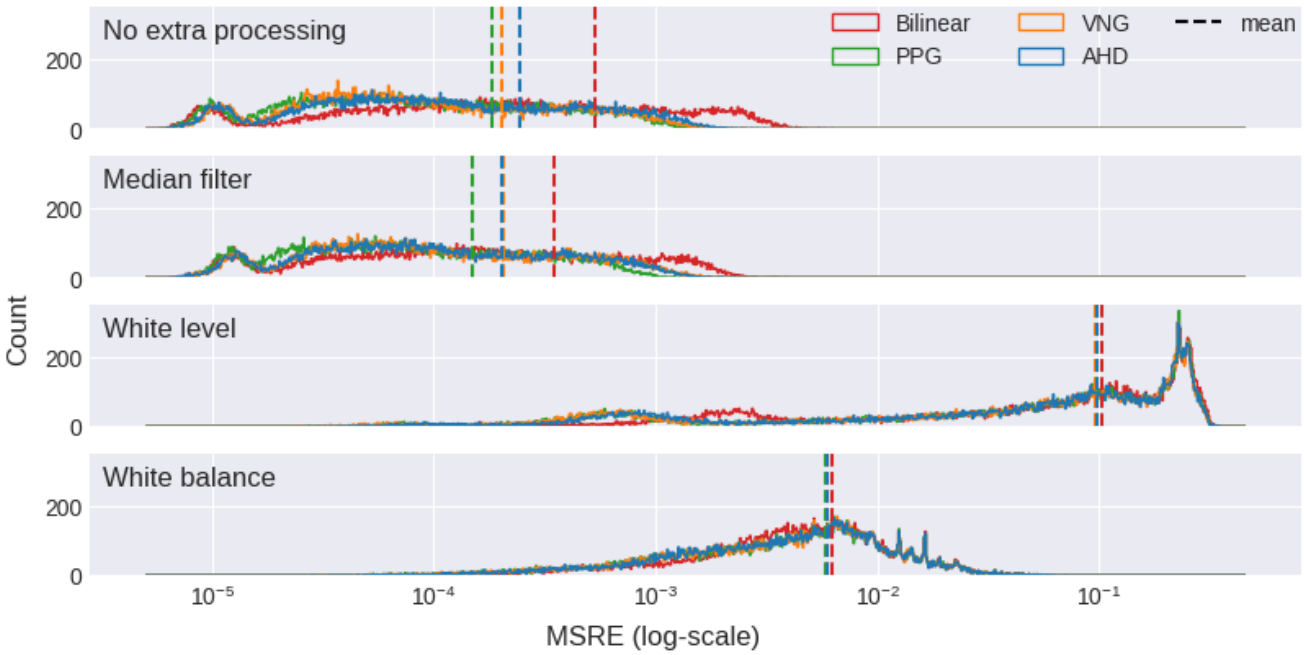

Figure 5.11: Histograms of MSRE between baseline signal and demosaiced signal for `EuroSAT`

For `EuroSat` the MSRE is lower for reconstructions using median filtering, than no processing. The white level processing produces an MSRE two orders of magnitude above the reconstruction without this processing step. White balance increases the distance by one magnitude.

	CFA	Bilinear	PPG	VNG	AHD
Demosaicing	-	$5.3 \cdot 10^{-4}$	$1.8 \cdot 10^{-4}$	$2.0 \cdot 10^{-4}$	$2.4 \cdot 10^{-4}$
U-net models	$1.6 \cdot 10^{-4}$	$1.6 \cdot 10^{-4}$	$1.9 \cdot 10^{-4}$	$1.7 \cdot 10^{-4}$	$1.9 \cdot 10^{-4}$

Table 5.3: MSRE for U-Net models and demosaicing algorithms for `EuroSAT`

The CFA models MSRE is lower than bilinear, VNG and AHD demosaicing, and higher than PPG demosaicing, without extra processing (see figure 5.3). The model MSRE is lowest for CFA, Bilinear, and VNG data, and highest for PPG and AHD.

White level distribution

The MSRE histograms provides insight into how the signal reconstruction is energetically affected by processing. The white level processing depends on the white level distribution shown below in figure 5.12:

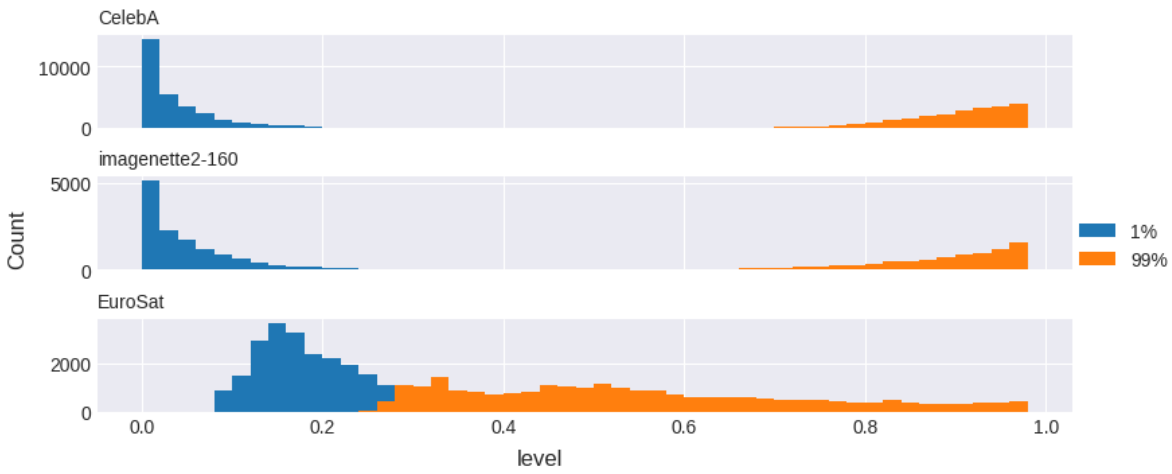

Figure 5.12: White level for `CelebA`, `Imagenette` and `Eurosat`

The white level distribution for the datasets that contains pictures is meant primarily for human viewing, `Imagenette` and `CelebA`, show the majority of the images use most of the dynamic range. While `EuroSat` does not use the entire dynamic range for the majority of the images.

Discussion

The MSRE of the U-Net models indicates that the MSRE of the input, compared to the original signal, does not predict the model performance. For each dataset, median filtering after demosaicing reduces the MSRE, however models trained on this data evaluated to a higher MSRE. This indicates information loss. This information could be used by the U-Net to reconstruct the signal.

Processing the image with white balance and white level, both increases MSRE for the demosaiced images and the models trained on these images. Given that the CFA capture has the same colour and luminance distributions, as the target signal, these kind of preprocessing can only increase the MSRE. The MSRE increase is greater if the target white level distribution varies greatly, as seen with `EuroSAT` in figure 5.11.

In this special case, using a camera, with a correctly calibrated white level and white balance, models trained on demosaiced images with no further processing, sometimes achieve the same MSRE, for this U-Net.

6. *Conclusion*

Experiment scenario 1, training denoising models

For a common use case, the image processing done to the images, specifically white level processing, reduces data fidelity when training the U-Net in our experiments, leading to significantly higher MSRE in most noisy and low-noise settings. For this case, we consider our hypothesis to be confirmed, MSRE is lower when training on CFA data, than on demosaiced data.

Removing individual processing steps for gamma correction and median filtering, reduces model MSRE performance. But this does not eliminate the MSRE gap between learning from CFA and learning from RGB data. For this case we consider our hypothesis confirmed, MSRE is lower when training on CFA data, than on demosaiced data, when comparing evaluation results for our specific U-Net architecture.

Removing the automatic white level setting, reduces the MSRE difference between models trained on CFA and AHD demosaiced data. Furthermore using bilinear demosaicing as a baseline shows that most of the MSRE differences aren't due to non-linearity in the AHD demosaicing algorithm. Part of the hypothesis is that non-linear processing, can reduce information and thereby reduce data fidelity. This experiment shows that the non-linearity in the demosaicing algorithm at most account for very little difference in model MSRE performance.

The biggest increase in model MSRE doesn't come from demosaicing in our experiments, indicating the other processing steps decrease the data fidelity more.

Experiment scenario 2, training signal reconstruction models

When training models on demosaiced data, that has not been processed further, it is possible to observe the same MSRE for these models, when comparing to models trained on CFA data. This effect is present for multiple different datasets. For this case we do not consider our hypothesis to be confirmed. Since MSRE is generally not lower when training on CFA data, than on demosaiced data, when compared to evaluation results for our specific U-Net architecture.

When training models on demosaiced data that has been processed further, our experiments show that MSRE increases for these models, when comparing to models trained on CFA data. This effect is present for multiple different datasets. The effect is present for all tested processing steps, both median filtering, white level setting, white balance, and combinations thereof. This indicates that further processing decreases data fidelity, which has an effect on our ML models, in these cases we consider our hypothesis to be confirmed, but limited. The limitation is, that the CFA data is generated by perfectly sampling from the RGB images, which does not reflect the imperfections typically found in real-world CFA data. This limits the use of the white level and white balance algorithms, since they are meant to correct conditions not present in our CFA data. Thus this scenario is applicable only for perfectly calibrated cameras.

The results also show that this U-Net approach does not constitute a general replacement for demosaicing, since performance in this regard is mixed. However this was not the goal or hypothesis.

The baseline performance evaluation, shows that processing can decrease MSRE toward the target signal, but increase MSRE model performance, as seen in the case of median filtering. This indicates, but does not confirm, that some processing for human viewing in non-linear ways, decreases the information in the data.

Bibliography

[1] R. Couturier, G. Perrot, and M. Salomon, "Image denoising using a deep encoder-decoder network with skip connections," in *Neural Information Processing*, Cham: Springer International Publishing, 2018, pp. 554–565. DOI: 10.1007/978-3-030-04224-0_48.

[2] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, eng, 2nd Edition. Newark: John Wiley & Sons, Incorporated, 2006. DOI: 10.1002/047174882X.

[3] C. Boncelet, "Chapter 7 - image noise models," in *The Essential Guide to Image Processing*, A. Bovik, Ed., Boston: Academic Press, 2009, pp. 143–167. DOI: 10.1016/B978-0-12-374457-9.00007-X.

[4] *imnoise - add noise to images*. [Online]. Available: https://www.mathworks.com/help/images/ref/imnoise.html#d126e175331 (visited on 04/19/2024).

[5] *skimage.util - general utility functions*. [Online]. Available: https://scikit-image.org/docs/stable/api/skimage.util.html (visited on 04/19/2024).

[6] C.-k. Lin, *Pixel grouping for color filter array demosaicing*. [Online]. Available: https://web.archive.org/web/20070326041737/http://web.cecs.pdx.edu/~cklin/demosaic/.

[7] E. Chang, S. Cheung, and D. Y. Pan, "Color filter array recovery using a threshold-based variable number of gradients," in *Sensors, Cameras, and Applications for Digital Photography*, N. Sampat and T. Yeh, Eds., International Society for Optics and Photonics, vol. 3650, SPIE, 1999, pp. 36–43. DOI: 10.1117/12.342861.

[8] K. Hirakawa and T. Parks, "Adaptive homogeneity-directed demosaicing algorithm," *IEEE Transactions on Image Processing*, vol. 14, no. 3, pp. 360–369, 2005. DOI: 10.1109/TIP.2004.838691.

[9] J. Liu, C.-H. Wu, Y. Wang, *et al.*, "Learning raw image denoising with bayer pattern unification and bayer preserving augmentation," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2019, pp. 2070–2077. DOI: 10.1109/CVPRW.2019.00259.

[10] H. Malvar, L.-w. He, and R. Cutler, "High-quality linear interpolation for demosaicing of Bayer-patterned color images," in *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 3, 2004, pp. iii–485. DOI: 10.1109/ICASSP.2004.1326587.

[11] T. Brooks, B. Mildenhall, T. Xue, *et al.*, "Unprocessing images for learned raw denoising," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 11 028–11 037. DOI: 10.1109/CVPR.2019.01129.

[12] Z. Li, M. Lu, X. Zhang, *et al.*, "Efficient visual computing with camera RAW snapshots," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 4684–4701, 2024. DOI: 10.1109/tpami.2024.3359326.

[13] C. Kantas, B. Antoniussen, M. V. Andersen, *et al.*, "Raw instinct: Trust your classifiers and skip the conversion," in *2023 IEEE 6th International Conference on Pattern Recognition and Artificial Intelligence (PRAI)*, 2023, pp. 456–460. DOI: 10.1109/PRAI59366.2023.10332021.

[14] C.-H. Liang, Y.-A. Chen, Y.-C. Liu, *et al.*, "Raw image deblurring," *IEEE Transactions on Multimedia*, vol. 24, pp. 61–72, 2022. DOI: 10.1109/TMM.2020.3045303.

[15] F. Altmann and G. Mark-Hansen, *cfa-raw*, 2024. [Online]. Available: https://github.com/GustavMH/cfa-raw (visited on 06/08/2024).

[16] Adobe Systems Incorporated, *Digital negative (DNG) specification*, Version 1.7.1.0, 2024. [Online]. Available: https://helpx.adobe.com/content/dam/help/en/photoshop/pdf/DNG_Spec_1_7_1_0.pdf (visited on 06/03/2024).

[17] "Information technology - digital compression and coding of continuous-tone still images: JPEG file interchange format (JFIF)," International Telecommunication Union, Geneva, CH, Standard, 2011.

[18] *OpenCV: Color space conversions*, 2024. [Online]. Available: https://docs.opencv.org/3.4/d8/d01/group__imgproc__color__conversions.html (visited on 06/09/2024).