UTRECHT UNIVERSITY

Faculty of Science

M.Sc. Artificial Intelligence thesis

Toward Computationally Efficient Real-World Raw Image Denoising with Knowledge Distillation

Supervisor:
J. Zhang

Candidate:
S.C.M. Rutten

First examiner:
R.W. Poppe

Student number:
6616089

Second examiner:
I. Önal Erugrul

In cooperation with:
Bosch Security Systems BV

June 28, 2024

Abstract

Deep-learning based denoising models are used to replace traditional denoising methods because of their better generalization ability and accuracy. Generating realistic pair wise data is important for the accuracy of these deep denoising models on real-world noisy sceneries. Most deep denoising works are focussed on the accuracy of the model, not taking the efficiency into account. Transformer models are the state-of-the-art performing denoising models. These transformer models are computationally too heavy for real-time denoising. Knowledge distillation can be used for compressing these models without losing much of the accuracy performance. We show that training deep denoising models on real-world noise model image pairs results in a good performance on the generated test set, and on real sensor noise image. Further, we show that the teacher-student architecture with knowledge distillation improves the accuracy of the student network. These student models gain a lot of efficiency without losing much of the teacher model accuracy, creating a better efficiency-accuracy trade-off for real-world image denoising.

Keywords - Denoising, Computer vision, Deep learning, Vision transformers, Knowledge distillation, real-world noise generation.

Contents

1. Introduction

Image denoising is part of the traditional image signal processing pipeline (ISP) [1] and refers to the process of reducing or removing noise from a signal or image, resulting in a cleaner and more visually appealing output. It is one of the oldest and most studied problems in image processing. Denoising helps high-level vision tasks, like image classification, improve their performance [2]. Despite the amount of research and well-performing traditional denoising algorithms, the challenge of preventing over-smoothing and blurring on real noise images remains [3]. These challenges paved the way for deep learning methods to be introduced to denoising. Deep denoising models can adapt more easily to the variable noise distribution. Transformers are state-of-the-art denoising models. The well-performing transformers are computationally heavy, making it impossible for real-time image denoising on embedded devices. That is why we aim to compress a transformer network. We try to reduce the accuracy gap between a big transformer and a small transformer by applying knowledge distillation.

For training such a model, a big dataset with realistic noise image pairs is required. To generate the dataset, we propose a pipeline to unify publicly available data sets. The public data sets contain various light conditions. To extend the dataset, we will use an unprocessing pipeline from [4] which enables us to convert sRGB images back to raw. Denoising low-light images is considered more challenging than good-light images because they have a higher level of noise. A realistic noise model is used to create the pair-wise data for the big dataset [5]. We make a selection of the datasets by training a deep denoising model on each separate dataset and test the performance. The selected dataset is then used to train the models and test the models.

The remainder of the chapter will provide an introduction to the study by first discussing the motivation and context, followed by the research goal along with the research questions, and lastly, the outline for the rest of the

paper.

1.1 Motivation and Context

The traditional ISP needs heavy tuning to process raw images into good, clean sRGB images. State-of-the-art traditional denoising methods tend to struggle with over-smoothing and blurred results, especially on real noise images [6]–[9]. The better performing traditional methods are computationally inefficient [6], [8], [10]. Replacing the traditional image processing blocks with a convolutional neural network (CNN) model has shown very promising results [11]. CNNs have shown to be better at adapting to variable distributed noise. The CNNs are integrated into the new surveillance cameras with system-on-chips (SOCs). These embedded neural network engines are becoming more powerful, enabling the use of deep learning methods for performing image processing tasks.

Deep neural networks tend to perform better on raw images than on sRGB images because they contain more information and have not undergone any compression or processing based on the noise [12]. Therefore, the deep learning focus tasks for our research are raw denoising and demosaicing. The denoising and demosaicing tasks are combined into one model to make it computationally more efficient [12]. A lot of well-performing deep denoising models are computationally too expensive for real-time denoising on embedded devices. Most denoising research focuses more on model accuracy than the balance between efficiency and accuracy [13], [14]. Finding the right balance between efficiency and accuracy is a hard challenge. The focus of our research is exploring computationally efficient deep denoising models for raw denoising and demosaicing. Knowledge distillation is a proven method for compressing deep learning models and keeping most of the accuracy [14], [15].

Deep learning models need large amounts of training data to reach sufficient performance. The training data needs to cover various conditions and environments to let the model learn how to operate in various scenarios. Capturing raw real noise image pairs requires editing the exposure time and

calibrating the camera while capturing image pairs. Changing the exposure time during the image capturing causes a slight time difference between the image pair, resulting in motion blur by the camera shaking or subject motion. Aligning these images is a hard and time-consuming challenge.

Therefore, most image pairs are generated by adding synthetic noise with a constant distribution to solve the challenges of capturing real image pairs. White Gaussian noise is the most commonly used type of synthetic noise [16]–[23]. However, real noise does not exhibit this constant distribution, instead it possesses a spatial correlation. The difference between the noise distributions result in deep denoising models not being effective in real-world applications [12]. To improve the real noise performance, a physics-based realistic noise model can be applied to clean images to generate realistic image pairs [5].

Using the more realistic image pairs results in a better performance for the deep denoising models on real-world noise applications [12], [24], [25]. For the clean images, public available raw data can be used. However, the images from different datasets are taken with different cameras and each camera has its properties, such as sensor type, file type, acquisition conditions, and type of image processing applied before storage, among others. Therefore, it is necessary to have a data processing pipeline to unify datasets. In addition, sRGB images are converted back to raw images using an unprocessing pipeline [4] to enable more data to be added to the unified dataset. After converting the sRGB images back, we use the same pipeline as the original raw images for the unification.

1.2 Research Goal

To efficiently reduce real sensor noise, we question if using knowledge distillation for transformer model compressing benefits the denoising performance of the more computationally efficient model. In this research, we aim to answer the following questions:

1. *What is the impact of compressing a transformer model for real world low-*

light image denoising using knowledge distillation? The teacher model will be shrunken to create a good balance between performance and computational efficiency. We would like to measure the impact on the performance of shrinking the deep denoising by answering the following questions:

(a) *How much knowledge is lost after compressing the model using knowledge distillation?* The performance difference between the teacher and the student model on the unified test set is compared using the quality matrices peak signal-to-noise (PSNR) and structure similarity index (SSIM). In addition, the student and teacher models are tested on the target sensor test set. This test set has no ground truth images, making it impossible to calculate the PSNR or SSIM. Therefore, the result images will be compared side by side.

(b) *How much computing power is saved after compressing the model using knowledge distillation?* The difference in computing power between the teacher and the student model will be measured by the average number of FLOPS used on a single test image from the target images.

(c) *What is the performance difference of the student network compared to the same network structure being trained from scratch?* Two identical network structures are trained differently. One is trained from scratch and the other is trained using knowledge distillation from the teacher network. The performance difference between the models is measured the same as described in research question 1a.

2. *How to generate pair-wise realistic noisy data on a large scale?* We unify publicly available datasets. We will add noise based on a real-world noise model, to generate pair-wise realistic noisy data. We want to test if this real-world noise model generates good enough image pairs for the deep denoising models to learn from. To help answer this question, we use the following sub-questions:

(a) *How to unify publicly available raw denoising datasets for the use of training a deep denoising model?* To test if the dataset is unified, we will train the model multiple times, leaving each time one dataset out of the training data. We will test the trained model on the test data of the missing dataset, computing the PSNR and SSIM. If the model performs well on different datasets, we could conclude that the dataset is unified.

(b) *What is the impact of the white balance and color correction matrix approximation for converting sRGB images back to raw images?* During the generation of the unified dataset, sRGB images are converted back too raw to have more training data. In the process of converting the sRGB images back to raw a few assumptions are made, we would like to know what the impact of these assumptions are on the model.

(c) *How large should the training dataset be for training the teacher network?* We create an extra dataset with by taking a random sample from the training data with only half the size. We train the teacher model twice, one time with the original training set and one time with the smaller dataset. Then we use the same test method as described in question 1a to conclude what the performance difference is for both models. We conclude if we need all data for training the teacher model based on the results. We repeat this process until we conclude that the performance from the model is significantly less than the performance of the model trained on the original training set.

1.3 Outline

The paper is organized as follows: in Section 2 the related work about image denoising and knowledge distillation is described by identifying key concepts and approaches. Section 3 presents the insights for the methodologies used for generating the dataset, implementing and testing the teacher denoising model, student denoising model and the knowledge distillation.

The results of the experiment will be discussed in Section 4. Lastly, the research questions will be answered to conclude the research in Section 5.

2. Related Work

First, this section describes the related work about traditional denoising in Subsection 2.1. Then in Subsection 2.2 deep learning for denoising is described. Further, the related work about knowledge distillation is described in Subsection 2.3.

2.1 Traditional Denoising

Traditional denoising is aiming to reduce noise by manipulating the pixel values based on the correlation between pixels in the original image. There are many well-performing traditional algorithms for this task [6], [26]–[35]. The first traditional methods were linear filters because of their mathematically simplicity [30], [31]. However, through their lack of performance, the linear filters have been replaced by non-linear filters [32]–[34]. Despite the amount of research and different traditional solutions, some challenges remain in the field of image denoising. In order to denoise the images, traditional denoising methods tend to blur the edges in the image and significantly reduce the level of detail [6]–[9]. Besides the loss of detail, traditional methods have a hard time performing well on images with a high level of noise due to over-smoothing [6]. Even if they would be able to denoise highly noised images, traditional methods assume that noise is homogeneous white Gaussian distributed, but real noise is more complex. The performance of the traditional methods drops significantly when tested on real noisy images [7]–[9].

The field of image denoising has been significantly improved on those challenges with the introduction of Bayesian least squares - Gaussian scale mixture (BLS-GSM) [36]. BLS-GSM is a hybrid denoising algorithm combining Bayesian estimation principles with a Gaussian scale mixture model. BLS-GSM creates image patches to operate on. For each patch, BLS-GSM

uses Bayesian estimation to estimate the clean patch and noise variance. The estimated noise component is subtracted from the noisy patch to produce the denoised image. The image is reconstructed by averaging the overlapping pixels from the patches [36]. Block-matching and 3D filtering (BM3D) algorithm [35] made another improvement in the denoising field at a later stage. BM3D has gained popularity due to its ability to effectively remove noise while preserving important image details. Unlike traditional linear filters, BM3D exploits the redundancy present in natural images by grouping similar image patches into 3D data arrays. By applying collaborative filtering techniques, BM3D can achieve superior denoising performance compared to linear filters and is often used to represent traditional models in a performance comparison [35], [37]. Therefore, some researchers focused on exploring and improving the non-linear filtering approaches like BM3D for better image denoising results [38]–[40]. Because BM3D is complex to implement, principal component analysis with local pixel grouping (LPG-PCA) [41] was published. LPG-PCA applies local pixel grouping (LPG) on the image patches to guarantee that only the sample blocks with similar contents are used in the PCA estimation. The LPG-PCA iterates twice over the image to reduce more noise [41]. These more complex denoising techniques that exhibit improved performance of the remaining challenges tend to incur high computational complexity and costs [6], [8], [10].

2.2 Deep Learning for Image Denoising

The complexity and lack of flexibility for well performing traditional denoising methods prompted the popularity of deep learning methods in the field of image denoising. Deep learning provides the availability of flexible performances with less computing power [11]. CNN-based image denoising enables adaptability to the specifics of the image. As a result of this newfound understanding, denoising emerged as the forefront of image processing research [42], [43]. There are three types of noise used for denoising images: simple synthetic noise models (mostly white Gaussian noise), real noise and realistic synthetic noise models [5].

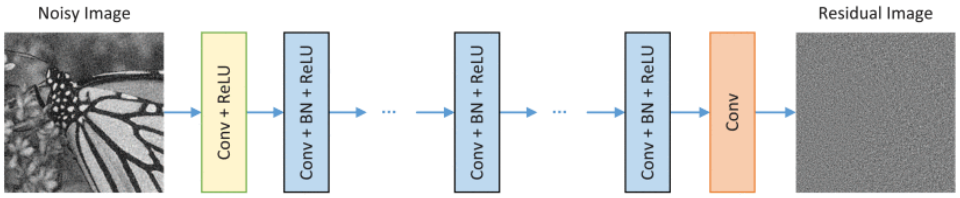


Figure 2.1: DnCNN architecture [16]

2.2.1 Deep Learning for Image Denoising using White Gaussian Noise

DnCNN was the first research to apply a CNN on a range of Gaussian noise levels, rather than focusing on one noise level [16]. The structure of the DnCNN network is shown in Figure 2.1. There are three types of layers used, marked with a different color in the image. The first layer (yellow) is a convolutional layer with ReLU activation to add non-linearity. The second until the second to last layers are similar to the first layer with additional batch normalisation [44]. The last layer is just a convolutional layer without ReLU or batch normalisation, to reconstruct the output. The depth of the network is task-specific. Higher vision tasks are often pixel-specific, therefore the reconstructed images from lower vision tasks should have the same size as the input image [45]. DnCNN adds zero padding to each layer to make sure that each feature map has the same size as the input image to remain the input image size through the whole model [16].

Not only improving the model performance, but also making them more computationally efficient, became a goal for image denoising. Therefore, in addition to a CNN network, CNBlind [17] was proposed to combine the CNN network with unsupervised learning that synthesizes training examples from specific noise models. Most of the CNN networks use back-propagation [46], [47] for training, however the proposed training method uses Stochastic online gradient learning. Stochastic online gradient learning randomly selects a small part of the training data and calculates the gradient for the first small network, starting with one hidden layer. After training the network for a set number of epochs, the weights of the first hidden layer are copied to a new network with two hidden layers. The weights from the hid-

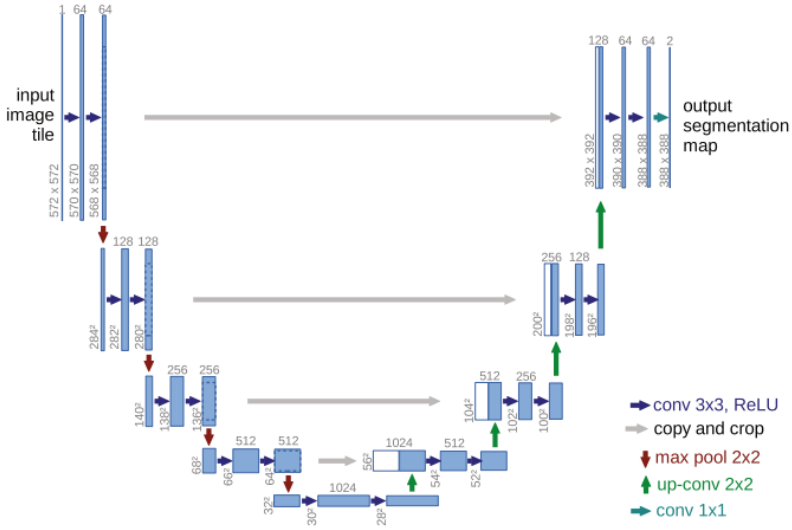


Figure 2.2: U-net architecture [18]

den layer to the output layer are discarded. The new network is trained for the same number of epochs as the old model, before repeating the procedure for N layers. The method allows big networks to train computationally efficient with matching their performance when trained with back-propagation [17]. With the same goal of making the training of models computationally more efficient, combined with the goal of training a model on very few training data, Ronneberger et al. [18] proposed the U-net model introducing skip connections to learn from smaller amounts of data. The architecture is shown in Figure 2.2. The architecture uses a feature contracting path and a symmetric expansive path. The contracting path is similar to the DnCNN network structure and repeats the structure of two 3x3 convolutions, followed by ReLU activation and 2x2 pooling, compressing the features for computational efficiency [19]. The expansive path upsamples the feature map by concatenation of a 2x2 convolution with the corresponding cropped feature map from the contracting path, followed by two 3x3 convolutions and ReLU. The model is trained with explained stochastic gradient descent method. The model was originally proposed for biomedical image segmentation but became a widely used model for image denoising, outperforming a lot of the previous models [15], [48]–[51].

The skip connections aim to recover spatial information lost during downsampling by skipping features from the contracting path to the expanding

path [52]. This is beneficial for denoising because the model has access deep into the model to features extracted early in the model without having to learn how to transfer them through the whole network, so the focus can be more on extracting new features. The skip connections are used in other structures as well, REDnet [20] implementing the skip connections to a more transparent model architecture in an attempt to improve the U-net performance. The idea of the encoder-decoder remains, but the architecture is not symmetric. The skip connections are used by adding the values instead of concatenating, adding the values occurs to an exponential grow of parameters while still receiving low-level features in deep layers [20]. Another network using the skip connections is MEMnet [21]. MEMnet implemented the skip connections to generate a memory block to increase the prior state influence on the outcome of the model. Additional to a feature contracting and an expansive path, they add multiple stacked memory blocks in between. The goal of this memory block is to learn which features are important for the reconstruction net to know. MEMnet has become less popular for denoising than U-net due to the simplicity and variety of available pre-trained implementations of U-net [53].

In 2015, Szegedy et al. [22] introduced a new strategy for CNN-based image classification. The transformer network became a very popular strategy within multiple research fields, among them NLP and image classification [54]–[56]. Transformers are also based on an encoder-decoder model. However, the difference with U-net is that U-net model is CNN-based, which is good for local feature extracting. Transformer models use self-attention, which is better for understanding broader context [57].

Due to their good performance on understanding the broader context, Chen et al. [23] decided to introduce the transformer network to the field of image denoising by proposing image processing transformer (IPT). The goal of the research is to improve image processing. Image denoising is only one of the subjects to achieve this. The model architecture is shown in Figure 2.3 and consists of four components: Heads, Transformer encoder, Transformer decoder and tails. The architecture is structured with multiple heads and tails to separate the different processing tasks, each task having one head

and one tail. The heads and tails consist of normal convolutional layers. The heads and tails are separated by a transformer encoder and decoder. The architecture of the transformer components are shown in Figure 2.4.

The transformer encoder first feeds each head to three distinct fully connected layers, forming the query, key, and value of a self-attention block. The queries and keys produce a matrix determining the focus of the input. This matrix is scaled down and SoftMax is applied to get the attention weights. The attention weights are multiplied with the values to compute the multi-headed attention output. This output is added to the original input and processed by feed forward layers. Here, another skip connection from before the feed forward layers is added. The summation is normalised to prevent gradient exploding. The decoder has similar sub layers as the encoder. The decoder is meant to regenerate the clean image [58]. The IPT model outperforms all discussed methods for image denoising [23].

IPT relies on over 115.5 M parameters and large-scale datasets (more than 1.1 M images) to achieve the good performance. Therefore, Liang et al. [59] introduced SwinIR. SwinIR is a transformer model with only approximately 10% of the parameters IPT has. SwinIR uses multiple Residual Swin Transformer Blocks (RSTB) containing multiple Swin Transformer Layers (STL) followed by a convolutional layer and an add-skip connection over all of them. The STL exist of a layer normalisation with an MSA, followed by a skip connection over them. This structure is repeated with an MLP instead of an MSA. Because of the recurrent architecture, SwinIR is easy to compress. Therefore, Liang et al. [59] propose lightweight SwinIR containing only four RSTB with 60 channels instead of six RSTB with 180 channels. This structure has even less parameters. Despite the more computationally efficient transformer networks, the downside for transformer models remains that they are very deep, with most of them being difficult to compress and they are computationally heavy.

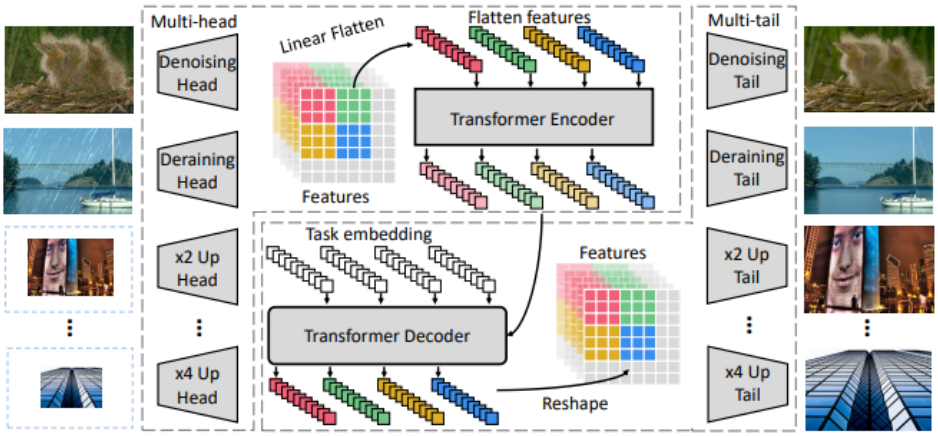


Figure 2.3: Transformer denoising [23]

2.2.2 Deep Learning for Image Denoising on Real-world Noise

Despite the powerful and good performance of the models, the training process is based on synthetic image data. Most data pairs are generated by applying white Gaussian distributed noise to the clean image. The white Gaussian noise model is not a good representation of real noise. Therefore, Plötz et al. [60] created a benchmark addressing the remaining challenges for real noise. For addressing these problems, they created the Darmstadt Noise Dataset (DND). They collected real noise images and (nearly) noise free images from the same viewpoint by changing the ISO values. In the research they find that, despite the performance of the CNN models on synthetic data, BM3D still has a better performance on denoising real data. The DND is created in normal light conditions, however denoising for low-light images is a big challenge as well.

Chen et al. [61] created a low light image dataset (SID) with the same technique of editing the ISO values. The input of the models is the raw image and the output is a sRGB image. This means that the model also includes demosaicing. They use CAN [62] and U-net [18] and train them on their collected data. They decide to exclude the residual connections from their research because they claim that the residual connections are not beneficial for their problem due to the different color spaces between their input and output. Both CAN and U-net are compared to the denoising of

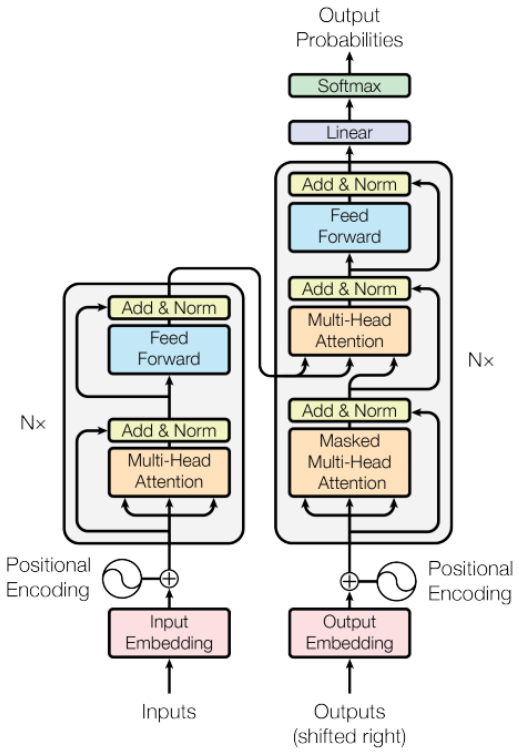


Figure 2.4: Transformer architecture [58]

a traditional pipeline and BM3D. From this research, we see that when the networks are trained on the real data they do outperform the traditional pipeline and BM3D. The U-net based model had a higher performance than CAN [63]. Anwar et al. [64] only focus on the denoising of sRGB images. Since the color spaces of their input and output are equal, they decided to include the residual connections and propose RIDNet. From their research we see that with similar color spaces it is beneficial to use the residual connections as they outperform previous models on sRGB to sRGB image denoising.

In Section 2.2.1 we saw that transformers have a good performance because they use self-attention, which brings the benefit of understanding the broader context. This helps for real noise denoising because it is less equally distributed than Gaussian noise. However, SwinIR restricts the context aggregation within the local neighbourhood due to the design choices, ignoring the main reason for using self-attention over convolutions. Therefore, Zamir et al. [65] propose a more efficient transformer than IPT, with the remaining benefit of understanding the broader context for the good perfor-

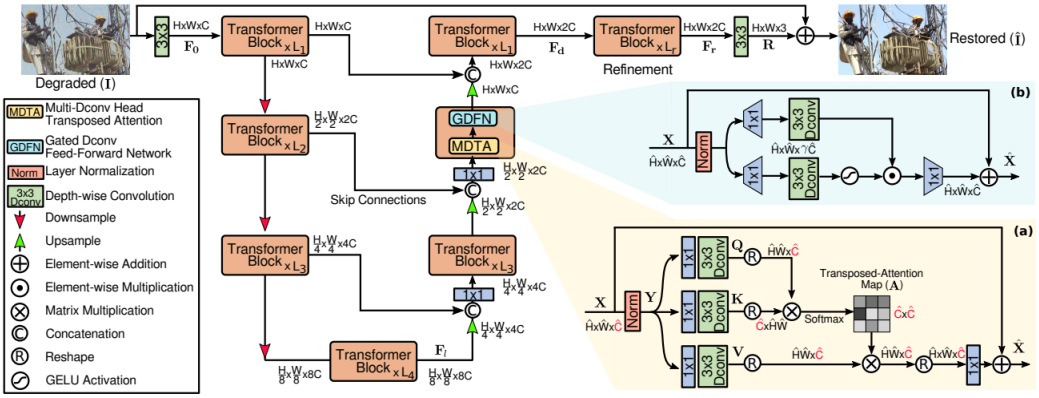


Figure 2.5: Restormer architecture [65]

mance on denoising real noise. The architecture of the Restormer network is shown in Figure 2.5. They use Transformer blocks in a similar down- and up-scaling approach as U-net. The transformer blocks contain a multi-Dconv head transposed attention (MDTA) block to compute query-key feature interactions across channels rather than spatial dimensions, followed by a Gated-Dconv feed-forward network (GDFN) to allow useful information to propagate further. Concatenation skip connections are used over the transformer blocks and an element wise skip connection from the input to the output to guide the lower extracted features through the network. They prove that with a lower number of FLOPS, the Restormer still outperforms IPT and SwinIR [65].

2.2.3 Deep Learning for Image Denoising on Improved Noise Models

Despite the improvement on real noise images, generating real noise pairs by extending exposure time causes motion blur due to camera shake or subject motion because it is difficult to align images captured using different exposure times. Therefore, Hasinoff et al. [12] introduced a new camera system to address these problems. With the camera system, they created the HDR+ dataset. To create the dataset, they use two streams. The first stream takes a picture and uses the normal ISP to generate the low-resolution image. The second stream takes multiple images with a consistent exposure time. These images are aligned and merged into a new and clean image.

With this dataset, they address the challenges of the blurred images in real world noisy datasets. They also claim that denoising from raw images is more efficient because the processing steps of the Image Signal Processing (ISP) pipeline have no influence on the image yet, and it leaves more bits per pixel [12], [25], [66].

Hasinoff et al. [12] were not the only ones trying to improve the real-world noise model. Wei et al. [5] proposed a sensor-specific physics-based noise formation model. They show that, particularly under extremely low light conditions, their model represents the real noise better. The general formula for digital sensor raw image D is:

$$D = KI + N,$$

where I is the number of photo electrons activated in the camera, K is the overall system gain and N represents the composed noise. The noise is sensor-specific. Because CMOS is currently the dominating image sensor, Wei et al. focus on the physics-based noise model. The noise is calculated with

$$N = KN_p + N_{read} + N_r + N_q.$$

K is the same factor as in the previous formula, the system gain. N_p is the photon shot noise, N_{read} is the read noise, N_r is the row noise and N_q is the quantization noise. In the research, they try to estimate the noise parameters by calibrating the camera on two images. The first image is captured from a white paper for uniformly light in the whole image, the second image is captured in a lightness environment with the shortest exposure time. In both images, there is no influence from the objects in an image but only the sensor input. The combination of the light and dark input results in the calibration of the noise parameters. The method is proven to estimate the noise better than the Gaussian model, and the pipeline is easier than

collecting real noise and clean image pairs [5]. However, the noise is often assumed to be stationary, meaning its statistical properties (such as mean and variance) remain constant over time or space. This simplification allows for easier analysis and modelling, which results in a lower accuracy than for models trained on physics-based noise pairs than on real-noise image pairs [67].

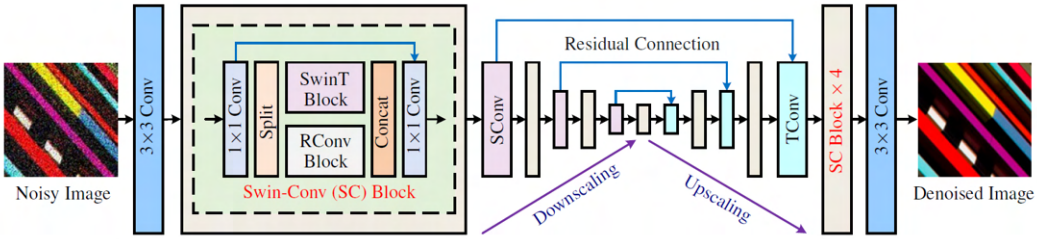


Figure 2.6: SCUNet architecture [68]

SCUNet [68] is a transformer model trained on an improved noise model. SCUNet is a combination of a CNN and a transformer, using the Swin Transformer Block from the SwinIR [54] and residual convolutional block from DRUNet [69]. The blocks are concatenated and combined with a 1x1 convolution layer into one Swin-Conv (SC) block [68]. By fusing these blocks, SCUNet aims to adopt the local model ability from the residual convolutional block with the non-local modelling ability of the Swin transformer block. The strided convolution (SConv) blocks are used for downscaling the feature maps, while the transposed convolution (TConv) blocks are used for upscaling. The SC Blocks with the up- and downscaling of the features are used in a similar U-net shape as Restormer.

2.3 Knowledge Distillation

For model compression, there have been many studies on pruning, quantization, low-rank decomposition and knowledge distillation. Knowledge distillation is a promising method for image denoising gaining efficiency without losing too much of the original accuracy [15].

2.3.1 General Knowledge Distillation Types

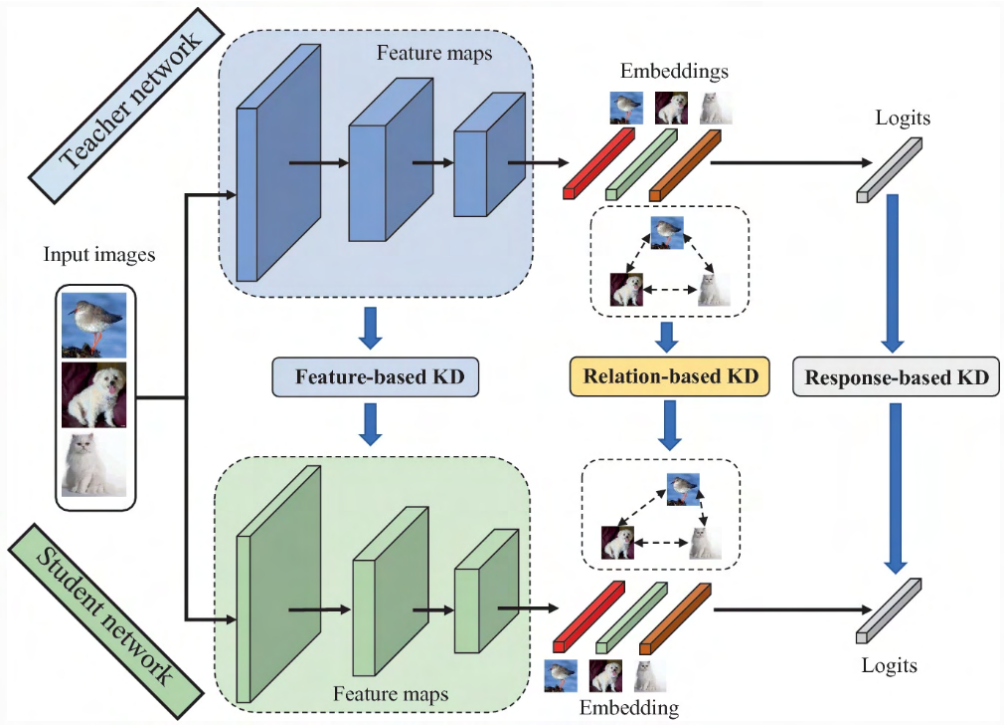


Figure 2.7: The schematic illustration of response-based, feature-based and relation-based offline KD between teacher and student networks [70]

Over the years, extensive research has been conducted on model compression. Model compression refers to the process of reducing the size and complexity of a machine learning model without significantly sacrificing its performance. Knowledge distillation [13] is successful in diverse fields including speech recognition, image recognition and natural language processing [14]. The idea of knowledge distillation is simple, to transfer knowledge from a pre-trained single large teacher network or set of large teacher networks to a single smaller, and computationally more efficient student network. Knowledge distillation has three different types: response-based knowledge, feature-based knowledge and relation-based knowledge. The different types are shown in Figure 2.7.

As the Figure 2.7 shows, response-based knowledge distillation focuses on the final output layer of the teacher and student model. The idea is that the student model learns from the prediction of the teacher model. The output knowledge is transferred by adding a distillation loss to the loss function used for training the teacher network. The teacher network loss is task-specific, the distillation loss is calculated between the teacher and student

output. The knowledge distillation loss is added to the original initiated loss function, giving the student loss function:

$$L_S = \alpha L_T + \beta L_{kd},$$

where L_t is the teacher loss, L_{kd} is the knowledge distillation loss and alpha and beta represent the trade-off between the knowledge distillation loss and the other losses. If the original loss function is combined with multiple losses, each function has a trade-off weight. The knowledge of the teacher network is also captured in the intermediate layers. Feature-based knowledge distillation aims to transfer the feature activation from the teacher to the student network by adding an extra distillation loss between the hidden layers, as shown in Figure 2.7. Relation-based knowledge distillation aims to transfer the relationship between feature maps to the student model. The relationship can be modelled as a correlation between feature maps. So far, there is only a little research using knowledge distillation for model compression on low-level vision tasks like denoising.

2.3.2 Knowledge Distillation for Image Denoising

Chen et al. [15] are the first to introduce knowledge distillation to the field of image denoising. They use a model-specific knowledge distillation method using feature-based knowledge distillation called Collaborative Distillation [3]. The method is based on U-net [18], also described in Section 2.2. The goal is to reduce the computational complexity from the original U-net so it can be deployed on embedded devices, like mobile phones. Collaborative Distillation was introduced in the field of style transfer. For style transfer, the decoder model is too task-specific to compress. Therefore, they focus on compressing the encoder. An encoder and decoder are tightly matched, introducing a new or compressed encoder to the same decoder would not work. Collaborative Distillation aims to transfer the knowledge of the current encoder E to a compressed encoder E' to make it compatible with the original decoder. Chen et al. [15] show that, despite a slight performance

decrease by compressing the encoder model four times, the proposed model still outperforms state-of-the-art models like noise2noise [71] and BM3D [35]. In addition, the proposed model has a significant computational advantage over U-net.

Young et al. [72] aim to improve their proposed denoising model using response-based knowledge distillation. They use the original U-net for the teacher network in their proposal of an adapted U-net-based model for the student network. For the proposed model, the convolutional layers from the original U-net are replaced with a variant of MobileNet-V2 [73] for efficient memory access. In addition, the skip-connection is shrunken by adding point-wise convolution. At last, they add a feature-align layer to make the model more aware of the input noise. They show that the performance of raw image denoising from their proposed model improves by using response-based knowledge distillation, which implies that using response-base knowledge distillation in beneficial for U-net based student models.

Li et al. [74] proposed a feature-based knowledge distillation method for an encoder-decoder based model with attention blocks. They are the only one so far bringing knowledge distillation to the field of denoising without using U-net as the teacher network. For the teacher and student model, they use the same framework but with different dimensions. The models contain two separate branches: the enhancement branch and the gradient branch. The gradient branch guides the network to retain more structural information. The enhancement branch is the main part of the network using an auto-encoder design. In between the encoder and decoder, they use Feature Attention Blocks (FAB), introduced by FFA-Net [75]. The features are transferred from the teacher network to the student network after each FAB to share the feature extraction knowledge. The teacher uses six FABs with 96 intermediate feature maps, while the student network only consists of three FABs with 48 intermediate feature maps, which reduces the used parameters. The distillation loss is calculated based on a normalised attention map Q to force the attention pattern of the student network to be as close as possible to the teacher network. Li et al. show that knowledge

distillation from an encoder-decoder network with one distillation channel between the attention blocks improves denoising sRGB images compared to the state-of-the-art methods [74].

Chen et al. [76] have an entirely different approach for applying knowledge distillation on U-net. They propose Two-Stage Raw Denoising (TSDN) using Expand-Shrink-Learning (ESL) to shrink the model. Instead of transferring the feature information for training the weights, they try to transfer the weights directly into a smaller model. ESL uses the Expand-Learning (EL) step to increase the model size remaining in the architecture. The expanded model is trained on the dataset from scratch. Subsequently, Channel-Shrink-Learning (CSL) and Layer-Shrink-Learning (LSL) are used to shrink the model to the target size. The CSL method halves the channel numbers of the middle layers and trains the model again on the model using the weights of the previous trained model as initialisation weights. Then the same steps will follow to the nearing middle layers. This is repeated until all channels are halved. After CSL, LSL will take place. LSL has a similar approach, but instead of halving the channel size, it cuts back the middle layers. After each cut, the model is trained again, using the weights of the previously trained model as initialisation for the weights of the shrunken model. TSDN outperforms U-net for extremely low-light conditions, while the model size is one-eight of U-net.

Knowledge distillation is not only used for model compression. Chi et al. [77] use two teachers with the same encoder-decoder structure as the student network. They claim that kernel prediction network has a high performance for constructing one clean image from a clean sequence input. A denoising network has a high performance in denoising a single image. Chi et al. [77] aim to combine the model qualities into one model using a motion and a denoising teacher. The knowledge is transferred after the encoder models using feature-based distillation to train the student network. With their research, they introduced multi-teacher learning to the field of denoising.

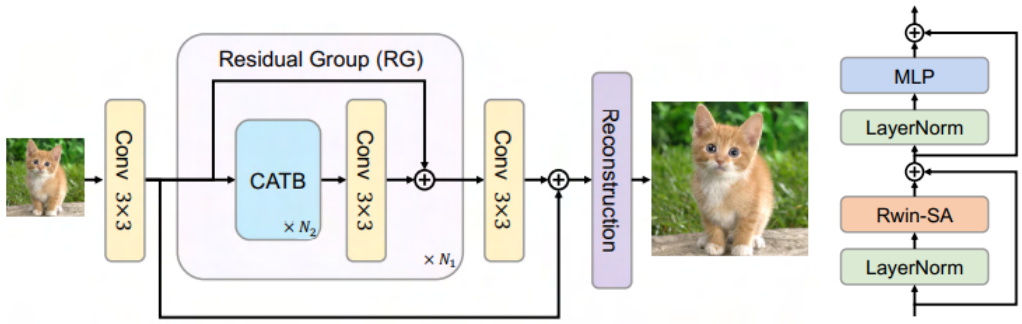
In Section 2.2 we saw that transformers have a high performance for

denoising. To the best of our knowledge, there has been no research for applying knowledge distillation using transformer models within image denoising. Because of the high performance of the transformer models, we would like to investigate the impact of shrinking a transformer model on the performance for denoising.

2.3.3 Knowledge Distillation for Transformers Replacing Image Processing Blocks)

Liang et al. [59] proposed a compressed version (lightweight SwinIR) in their original SwinIR research. The combination of SwinIR as teacher and lightweight SwinIR as student model is used in multiple knowledge distillation image processing research. Jiang et al. [78] propose a compressing model applied for super resolution using SwinIR and lightweight SwinIR. The method uses response-based knowledge distillation and pruning. They show that the performance of lightweight SwinIR, the compressed version of SwinIR, has only a tiny drop compared to the original SwinIR while it contains approximately 10% of the computing power. Xie et al. [79] also use SwinIR and lightweight SwinIR for their proposed knowledge distillation approach. They use a Frequency Similarity Matrix (FSM) and an Adaptive Channel Fusion (ACF) to transfer the relation-based knowledge from SwinIR to lightweight SwinIR. FSM is meant to extract frequency information from intermediate images to transfer the knowledge about edge and texture parts. ACF is feature-based and is meant to transfer the spatial information for up-sampling the images. They show that using relation-based knowledge distillation in the training process improves the performance of lightweight SwinIR compared to training the model from scratch.

For knowledge distillation, it is more efficient for the student and teacher models to have the same architecture framework, with different dimensions [80]. Models like Restormer because rely on a more complex structure, making it harder to compress the model (see Section 2.2.2). This could be a potential reason why, as far as we know, Restormer is not used for knowledge distillation so far. As described in Section 2.2.2, SwinIR uses local windows,



(a) CAT architecture **(b)** CATB

Figure 2.8: The architecture of CAT [81]

losing the transformer advantage of understanding broader context. Chen et al. [81] proposed CAT, a transformer model with recursive blocks aiming to effectively combine convolution with attention mechanism. They use horizontal and vertical rectangle window attention in parallel, using different heads for expanding the attention area. The architecture of CAT is shown in Figure 2.8. The dimensions of the CAT are variable by adapting the number of Residual Groups and the number of CATB in a Residual Group. This simplifies creating a teacher and student architecture with the CAT framework. They show that with six residual groups containing six CATB they achieve similar results to the Restormer model. Despite the potential, there has been no research using the CAT framework for knowledge distillation so far.

3. Method

In this section, we present the methodology used to develop our efficient transformer models. Section 3.1 gives a high-level overview of the different stages used to develop our model. The stages use the same ISP blocks, which is why Section 3.2 describes the methodology for those blocks. Sections 3.3, 3.4 and 3.5 will go into detail about the stages, offering a comprehensive understanding of our approach.

3.1 Overview

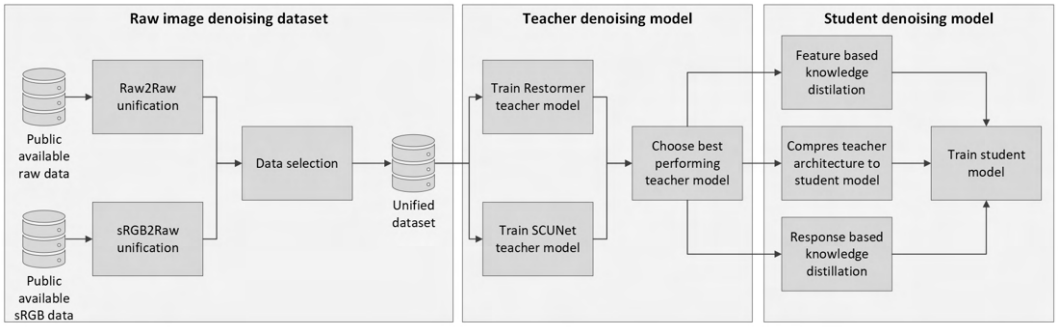


Figure 3.1: The schematic overview of the used methodology

In Section 1.2 we described that the goal for our project is to efficiently reduce real image noise using deep denoising. In Section 2.3 we saw that knowledge distillation is a well-proven method for training efficient models with high accuracy. To apply knowledge distillation, three stages are required. The stages are shown in Figure 3.1. The first stage is to create a pair-wise image dataset for denoising, consisting of a noisy image with the corresponding clean image. In Subsection 2.2.2 we saw that it is time-consuming to gather a high amount of pair-wise data containing the sensor specific real-world noise. In Subsection 2.2.3 we saw that improved noise models are more efficient for creating pair-wise images without losing too much of the real-world characteristics [5]. In this Subsection, we have also

learned that denoising is more effective from raw images than from already processed sRGB images [5], [12], [61], [66]. This is why we have decided to create a pair-wise raw image dataset by applying a real-world noise model on public available datasets.

The images from public available datasets are taken with multiple different sensors. These sensors have different specifications, resulting in different properties for the raw image. Most images are directly processed to the unified sRGB format [82]. However, in Subsection 2.2.3 we have seen that denoising is more effective on raw images. These raw images still contain the different sensor-specific properties, unifying those properties before training improves the model performance [12], [25]. Properties which should be unified are black level subtraction, Bayer pattern, rotation of the image and number of bits [25]. To unify the data, we designed a raw2raw pipeline to convert raw datasets to the unified format. The number of public available, high-quality raw image datasets is limited. The sRGB format is the standard color space for images on the internet [82], which is why the sRGB format is widely used and easily accessible. By incorporating sRGB images into our dataset, we can significantly increase the diversity and representativeness of the data, allowing our model to learn from a broader range of visual inputs and improve its performance in real-world scenarios. The input of our pre-processing pipeline is a raw format. Hence, we use an unprocessing pipeline converting the sRGB images back to raw [4]. These raw images have to have the same unified format as the other raw datasets to reach the optimal performance of the model. Therefore, the raw2raw pipeline will also be applied on the unprocessed images forming a sRGB2raw pipeline.

Learning from more data usually improves the model, as long as the data is good [83] (see Section 2.2). To reduce the training time as much as possible, we want to select only the high-quality datasets. It is too time-consuming to check the quality of the images manually, which is why we will apply an image quality assessment, training a deep denoising model on each separate dataset and compare the performance results [84]. During the image quality assessment, the model architecture will stay the same.

Therefore, the performance differences will be based on the features from the datasets. As long as the model is good enough to extract those features well from the data, the quality comparison with a different model will be similar to the quality comparison with the final model [84]. We will compare the images based on U-net [18] because it is a widely used model for image denoising, it is easy to set up, and it is computationally less heavy than transformer models [15], [19], [48]–[51] (see Subsection 2.2.1). After comparing the dataset qualities, we try to understand why certain datasets score better than others by using t-SNE [85]. These insights will allow us to identify the attributes that would be most beneficial if we were to add more data to enhance the model. We aim to get the insights by applying t-SNE because it is a popular method for exploring high-dimensional data [86], [87].

In Section 2.2 we saw that it is easier for a model to reach a high accuracy if efficiency is not considered. In Section 2.3 we saw that good performing models can help to guide more efficient models to a higher accuracy. Therefore, the second stage is to train a teacher model on the selected unified dataset from stage 1. The goal of the teacher model is to achieve the highest possible accuracy while ignoring the efficiency aspect. As we have seen in Section 2.2, transformers are high performing models for deep denoising, but they are not very efficient [23], [59], [65]. We are aiming to find the best performing denoising model among some popular choices in the image restoration domain. We have learned that different sensor-specific noise can affect the performance of a model (see Subsection 2.2.2). We chose to train multiple transformer models for the teacher model in order to experiment which performs best for the target noise. To the best of our knowledge, there are no widely used transformer models for raw image denoising. Therefore, we chose to adjust widely used transformers for sRGB image denoising. We selected the Restormer model [65] (See Section 2.3) and the SCUNet [68] for the comparison because they are well performing and widely used for real-world image denoising [68], [88]. More about training the teacher models is described in Section 3.4.

Looking back to the goal, we see that the teacher model does not fit the

requirement for being efficient (see Section 1.1). Therefore, the third stage is to train the student model. The goal for the student model is to be more efficient than the teacher model, without losing much of the performance. To achieve the low performance loss for the smaller model, we decided to use feature-based and response-based knowledge distillation (see Section 2.3). Knowledge distillation is a proven method for maintaining the model's performance while decreasing its size [13], [14]. Feature-based and response-based distillation are the most used methods within the field of low-level image processing [15], [79], [89]. It is beneficial for the student model to have the same type of structure as the teacher model [80] (see Section 2.3). Hence, we choose the best performing teacher model and compress that structure for the student model.

3.2 ISP Blocks

In this section, we will describe the methodology of multiple ISP blocks. These blocks are used throughout the different stages. First, we will describe white balancing in Section 3.2.1. In Section 3.2.2 we will describe color correction and in Section 3.2.3 we will describe about gamma correction.

3.2.1 White Balance

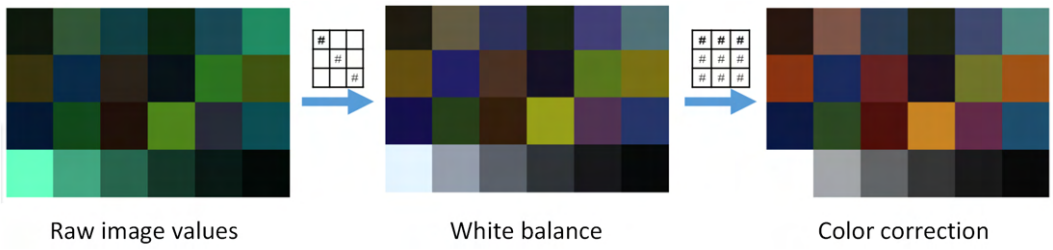


Figure 3.2: Color mapping from sensor spectral to CIE XYZ using white balance and color correction

Each camera sensor has its own spectral response, which creates differences in color matrixes of an image. This would result in different images if two cameras with a different sensor would make an image of the same setup. The color space of the raw image is mapped to a general color space

to prevent these color differences in the processed images. This process takes two steps, applying white balance and color correction. Figure 3.2 shows the color shift of the two color mapping steps.

White balance is used to adjust the temperature of the light in the image. With cold light, the color of white surface is more blue while with warm light, the color is more yellow. For automatic white balance, the algorithm needs to determine the sensor's response to the scene illumination. In the pipeline, we use the "Gray world" algorithm because it is the easiest method to apply [1]. For the raw image, it means that the white balanced image is calculated by the following formula:

$$\begin{bmatrix} B' \\ G' \\ G' \\ R' \end{bmatrix} = \begin{bmatrix} G_{avg}/B_{avg} & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & G_{avg}/R_{avg} \end{bmatrix} \begin{bmatrix} B \\ G \\ G \\ R \end{bmatrix}$$

White balancing the sRGB image has the same goal as for white balancing sRGB. The difference is that the white balance now needs to be applied on a 3-channel sRGB image. For the sRGB format, the white balanced is calculated by the following formula:

$$\begin{bmatrix} R' \\ G' \\ B' \end{bmatrix} = \begin{bmatrix} G_{avg}/R_{avg} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & G_{avg}/B_{avg} \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

The Gray world algorithm assumes that the average reflectance of a scene is gray. The color patch of the raw image is not relevant, the white balanced color patch for the same sensor is always similar (if the same white balance algorithm is used) [1].

3.2.2 Color Correction

The second step for the color mapping is color correction. The goal for color correction is to approximate the true colors based on the white balanced image. As described in Subsection 3.2.1, the color patches from the white balanced images are similar. Therefore, we can use a sensor-specific constant matrix for to convert the white balanced image to the true color image. For the unified dataset, we use the identity matrix because there are multiple sensors used with an unknown color correction matrix. For the target data, we use the following matrix:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1671 & -316 & -51 \\ -363 & 1024 & -369 \\ 43 & -386 & 1541 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

The color correction matrix is configured based on test images taken with the target sensor. The test images are taken from a Macbeth color cherker (see Figure 3.3).

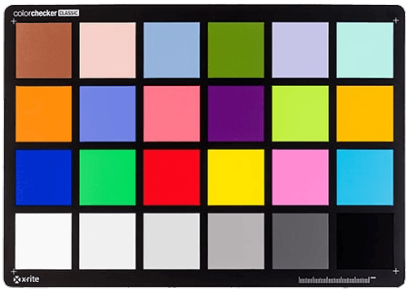


Figure 3.3: Macbeth color checker

A Macbeth color checker is a board with different precise colors with know RGB values. The color correction matrix is then configured by searching the color correction matrix which gives the least color deviation E on the Macbeth color checker.

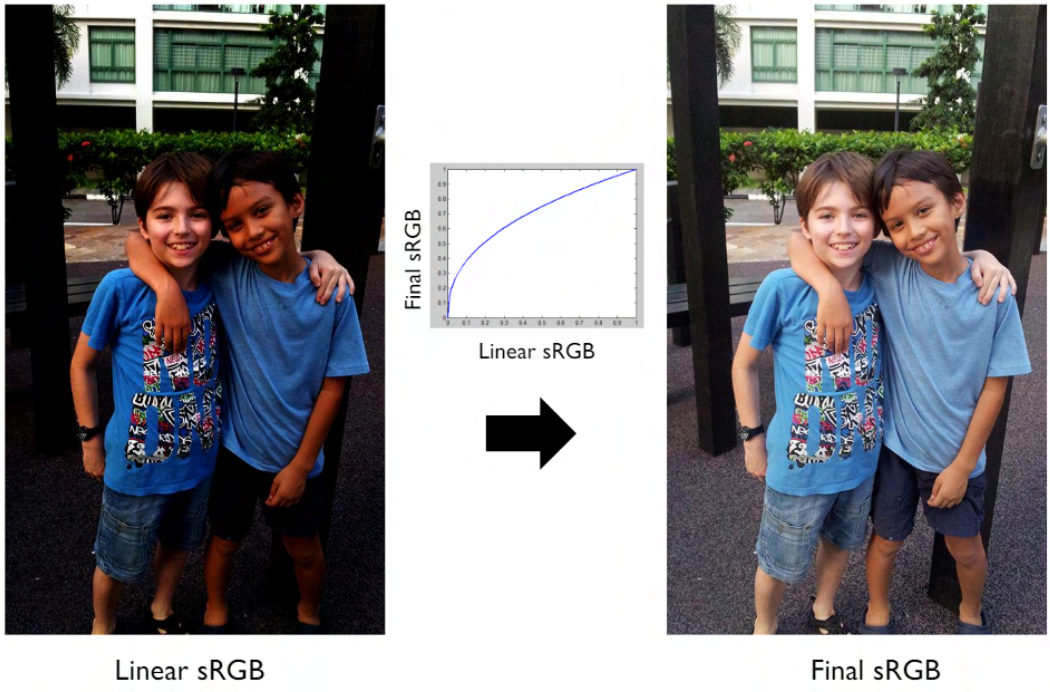


Figure 3.4: Gamma correction

3.2.3 Gamma Correction

The goal of gamma correction is to standardize the way colors are displayed and interpreted across different devices, such as computer monitors, printers, and digital cameras. Gamma correction can be applied to the raw image or to the sRGB image. The sRGB gamma curve is designed to ensure that colors are accurately represented and displayed consistently, regardless of the device being used. This helps to create a more uniform and predictable color experience for users across different platforms and devices. The effect of sRGB gamma is shown in Figure 3.4. The encoding of the sRGB gamma is related to Steven's power-law [90]. The following formula is used as sRGB gamma:

$$R'G'B' = (12.92 * RGB * ind) + (1.055 * RGB^{1/2.4} - 0.055) * (1 - ind)$$

Where RGB is the image and ind is an index matrix marking the darkest

regions of the image. The *ind* matrix is generated by $rgb < 0.0031308$. *ind* is introduced to brighten and create more visibility in the darkest parts of the image. This is important for the pipeline of the research because denoising is mainly targeted on low light images.

3.3 Raw Image Denoising Dataset

This section describes the methodology used to generate the pair-wise raw dataset for training the deep denoising models. First, we will describe the characteristics of the public available datasets in Subsection 3.3.1. These characteristics are important for the method to unify the datasets. How the datasets are unified is described in Subsection 3.3.2. From the unified data, we make a selection based on the quality of the images in Subsection 3.3.4.

3.3.1 Sensor-Specific Data Characteristics

Multiple different public available datasets are used for training the deep denoising models. Each dataset has its properties such as sensor type, file type, acquisition conditions, type of image processing applied before storage, among others. We use datasets with raw and sRGB formats. The raw formats are images directly taken from the sensor and differ based on the sensor they are taken with. These image sensors have different patterns to capture colors, named Bayer patterns [25]. Some examples of the easiest Bayer patterns are shown in Figure 3.5. The properties of the raw datasets are shown in Table 3.1.

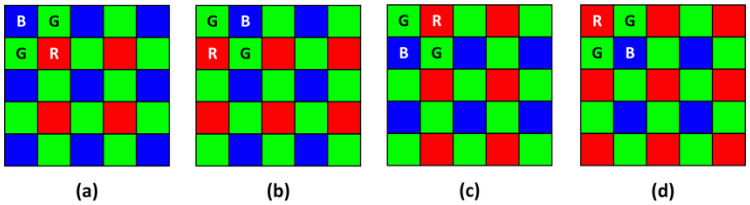


Figure 3.5: Bayer pattern examples

The differences in black level subtraction, Bayer pattern, rotation of the image and number of bits between the datasets should be unified to use the datasets consistently [25].

Dataset	Format	Light condition	Number of images	Image size	Bayer pattern	Number of bits	Black level subtracted
os04a10 (target)	Raw	Low/ med	300	1920x1080	BGGR	16	Yes
HDR+ [12]	Raw	Low	3640	4068x3036	RGGB/ BGGR	10-14	No
SIDD [66]	Raw	Low/ med/ high	160	5328x3000	RGGB/ BGGR/ GRBG	1	Yes/ No
SID [25]	Raw	Low/ high	231	4288x2848	RGGB	14	No
Raise [91]	Raw	Low/ Med/ High	8156	3008x2000, 4288x2848, 4992x3280	RGGB	12/14	Yes
Mit5k [92]	Raw	Low/ med/ high	5671	4368x2912	BGGR	16	No

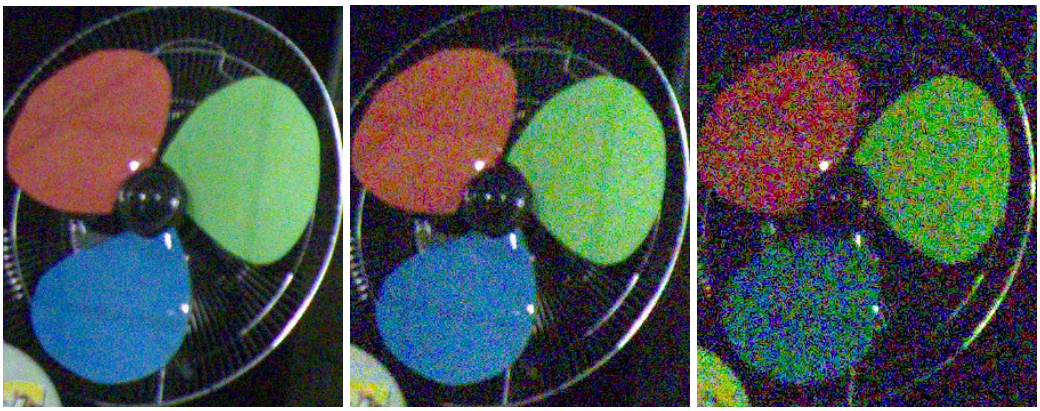
Table 3.1: Dataset details

The real noise data is in raw format. The characteristics from the noise data are shown in Table 3.1. The target sensor is an OS04A10 sensor. The real noise images are taken in a studio with different light conditions and outside with different noise levels. The noise range is from 32db until 63db, which is from low noise to very high noise. The examples for the noise levels are shown in Figure 3.6.

We also public available sRGB datasets for unprocessing. sRGB images are processed to the standard color space for images on the internet, which means that they have already processed properties like Bayer pattern and black level subtraction [82]. Table 3.2 shows the properties for the sRGB datasets.

Dataset	Format	Light condition	Number of images	Image size
Flickr2K	sRGB	High	2650	2040x1140

Table 3.2: Dataset details



(a) 32db **(b)** 45db **(c)** 63db

Figure 3.6: Real sensor input images

3.3.2 Data Unification and Preparation

This subsection describes how we unify the raw datasets in Subsection 3.3.2.1 and how we unprocess the sRGB images in Subsection 3.3.2.2.

3.3.2.1 Raw to Raw Data Unification

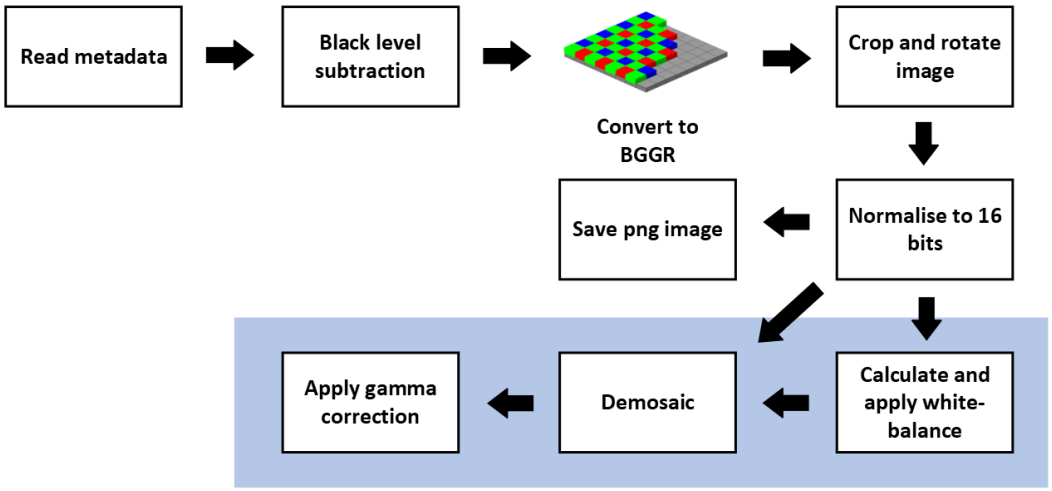


Figure 3.7: Raw to raw unifying pipeline

Figure 3.7 shows the overview of the raw2raw unifying pipeline. The first step of the pipeline is subtracting the black level. For pixels with "no light", the values should be zero. However, sensor noise can cause that those pixels still have a value above zero. Those values can be corrected by subtracting black level [1]. For the target data, the black level is already subtracted before saving the raw data. It is important to match the black

level settings from the unified dataset to teach the model how to handle zero values. For some datasets, the black level is already subtracted (see Figure 3.1). For the other datasets, subtracting the black level is the first step of the unifying pipeline.

The next step is unifying the Bayer pattern. Having the same Bayer pattern across the dataset is beneficial for the performance of the model [25]. Liu et al. proposed a method to unify the Bayer patterns of raw image data [25]. This process is shown in Figure 3.8. The target Bayer pattern for the dataset is BGGR. The images with a different Bayer pattern are cropped with one row from the top and bottom (GRBG to BGGR), one from both sides (GBRG to BGGR) or one from top, bottom and both sides (RGGB to BGGR).

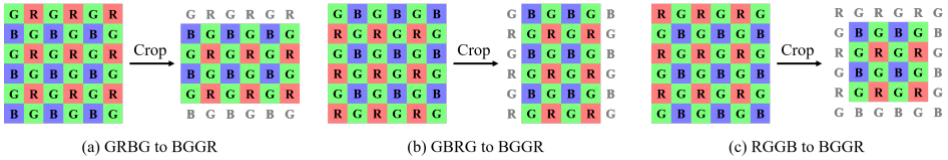
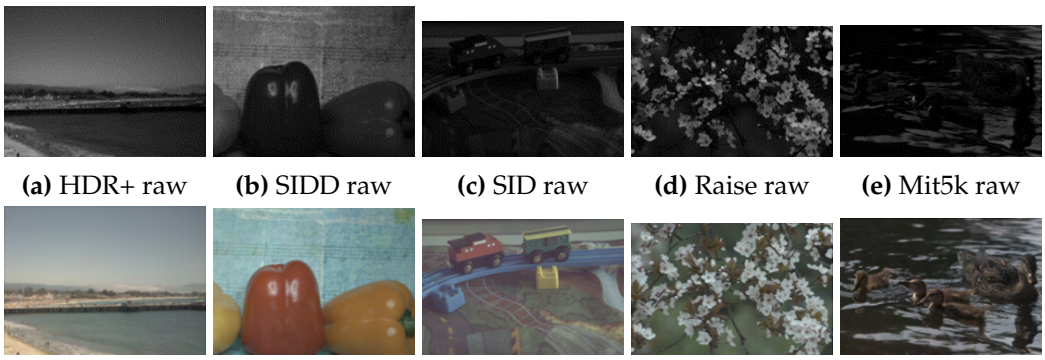


Figure 3.8: Bayer pattern unification [25]

After converting to the right Bayer pattern, the image size is reduced. The bigger the image is, the longer it takes for the model to denoise the image. Later we also see that the images are cropped for training in the training pipeline (see Section 3.3.3). Saving the unified images in a larger resolution would only be beneficial if multiple patches were created from the images, which is not the case. We therefore crop the images in advance to save disk space. For cropping, a random area with the fixed size is chosen. Then the image is rotated based on the metadata. Images can be taken with different positions of the sensor. The camera saves how the image should be rotated to have the image not upside down. We rotate the images based on the rotation detection of the camera because the target sensor also corrects the rotation before saving.

To normalise the image data, the number of bits is converted to the target of 16 bits and the dtype is set to uint 16. Normalising the data is important for the deep denoising performance [93]. With normalising the data, the

raw2raw unification is finished.



(a) HDR+ raw **(b)** SIDD raw **(c)** SID raw **(d)** Raise raw **(e)** Mit5k raw

(f) HDR+ sRGB **(g)** SIDD sRGB **(h)** SID sRGB **(i)** Raise sRGB **(j)** Mit5k sRGB

Figure 3.9: RAW2RAW image validation

After saving the file, we want to validate the unified format for each dataset. For validation, we can visualise the unified images. For these visualisations we will add white balance, demosaic the image and add gamma correction because these steps highlight the effect of the raw2raw unification pipeline. The steps are shown in Figure 3.7 The results are shown in Figure 3.9, the upper image is from the HDR+, the second image from Flickr2k and the last image is from the Mit5k dataset.

3.3.2.2 sRGB to Raw Data Unification

For the unified dataset, two sRGB datasets are used, being Flickr2K and Unsplash. The properties of the datasets are shown in Table 3.1. To unprocess the sRGB data back to raw, the proposed method from Brooks et al. [4] is used. This is a widely used pipeline for image unprocessing [94], [95]. The steps are shown in Figure 3.10.

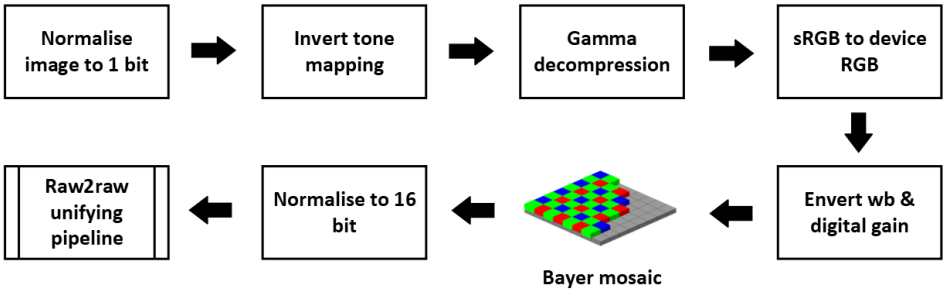


Figure 3.10: Unprocessing pipeline [4]

For Flicker2K there is no metadata available, the normal distributed color correction matrix, digital gains and white balance will be used to reverse the processing steps. The results are shown in Figure 3.11.

(a) Flickr2k sRGB source image

(b) Flickr2k unprocessed raw image

(c) Flickr2k sRGB (verification)

Figure 3.11: sRGB unprocessing and unification results

The restored image in 3.11c is a bit more greenish than the source image in 3.11a because color correction is not yet applied to the image. This is not necessary for validation of the unification steps.

3.3.2.3 Adding Noise Based on Real-world Noise Model

The unified datasets contain only clean images. For training the model, clean and noisy image pairs are required. Therefore, a real-world noise model is used to generate realistic noise for adding to the clean images. The real-world noise model is an extension of the Gaussian noise model and is formulated as:

$$Noise = N(0,1) * \sqrt{\frac{Aa * Ad * I}{Nsat} + Ad^2 * (Aa * CT1n * CT2n)^2 + (PRNU * s)^2}$$

Where $N(0,1)$ is the normal distribution with $\mu = 0$ and $\sigma = 1$. Aa is the analogue gain. Ad is the digital gain. I is the normalised image before adding the noise. $Nsat$ is the saturation limit, determined by exposing the sensor long enough that every pixel reaches the full capacity. $PRNU$ is the per pixel gain. $CT1n$ and $CT2n$ are respectively $STD(Nt1)$ and $STD(Nt2)$ where $Nt1$ and $Nt2$ are components of the thermal noise before and after the analogue gain. $CT1n$ and $CT2n$ are determined by capturing sensor

output in the dark at various analogue gain settings. The read noise is then calculated for each gain setting. A fit is performed between the calculated read noise and sigma according to the noise model to determine CT1n and CT2n.

The amount of noise is controlled by calculating the analogue gain Aa and digital gain Ad. Noise is often displayed in the dB format. First, we need to calculate the linear gain from the dB using the standard formula:

$$Al = 10^{\frac{db}{20}}$$

From the linear gain, we can calculate the sensor gain with the following formula:

$$As = \frac{Al}{Ac}$$

Where Ac is conversion gain, which is 4.1433 for the target sensor. With the sensor gain, we can determine the analogue gain. Usually, the analogue gain is measured from the sensor. However, the analogue gain can never be higher than the total sensor gain. Therefore, if the sensor gain is lower than the maximal analogue gain, we set the sensor gain as analogue gain. Giving the formula:

$$Aa = min(As, Aa_{max})$$

From the analogue gain and the sensor gain, we can calculate the digital gain using:

$$Ad = \frac{As}{Aa}$$

After calculating the noise, it is added to the image. The image with the simulated noise can reach outside the image range. Therefore, an important step after adding the noise is to clip the image. These steps result in the following formula:

$$In = clip_{0,1}(I + Noise)$$

3.3.3 Pre- and Post-Processing Pipeline

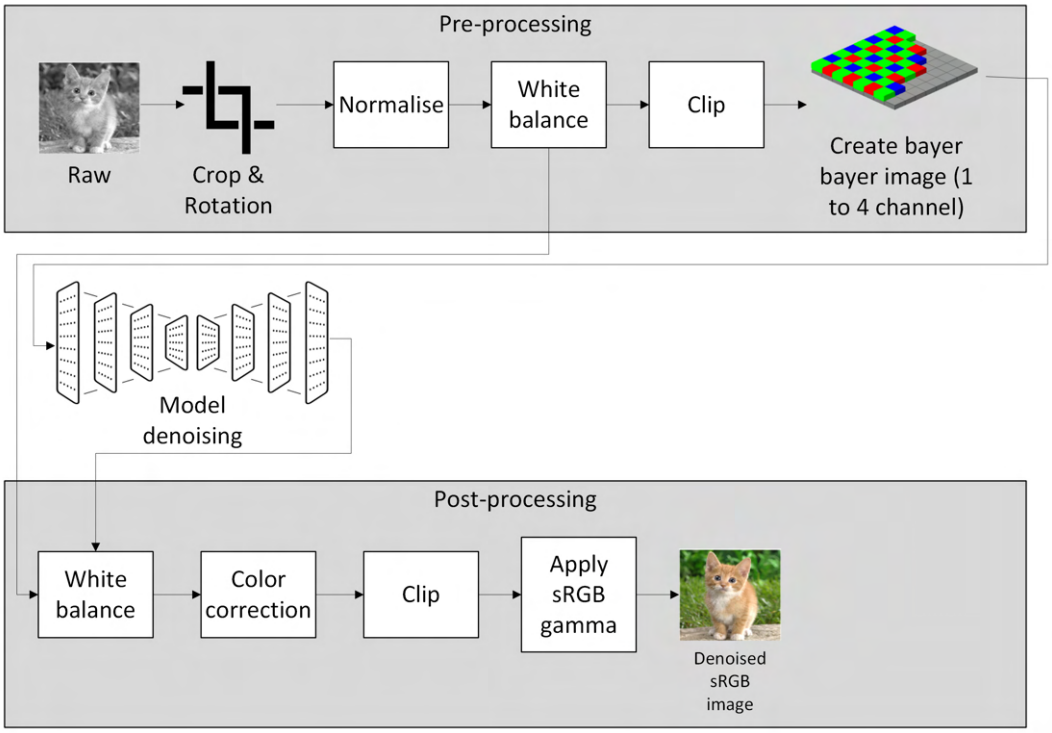


Figure 3.12: Overview of the data process pipeline for denoising

This section describes the details about the pre- and post-processing pipelines, forming an ISP. The overview of the ISP used for the experiments is shown

in Figure 3.12. The first step of the ISP is to crop the image, this step is only used for training the model with the purpose to speed up the training process. After cropping, the image is normalised to 1 bit. The next step is to white balance the raw image. White balancing, important for the generalisation of colors between different sensors. Applying white balance is based on the color values of an image, the colors from the raw image are saved for later. More about white balance is described in Subsection 3.2.1. Applying white balance can result in values outside the value range, which causes artefacts in the image. Therefore, it is important to clip the image as the next step. The last step is to extract the color channels from the images according to the Bayer pattern. This makes it easier for the model to handle the different colors.

The pre-processed image is then used as input for the denoising model. The denoising model takes a 4-channel raw image as input and outputs a 3-channel RGB image. After the model prediction, the post-processing starts. The first step of post-processing is white balancing the generated image. The goal to white balance again is to get the same white values as the original image. To reach this goal, the measured colors from the first white balancing step are used. White balance for the sRGB image is described in Subsection 3.2.1. The next step is to apply color correction to balance the colors in a digital image. More about color correction is described in Subsection 3.2.2. Similar as we have seen in processing, the next step is to clip to make sure the image values are within the range after adding values. The last step is to apply gamma correction, which makes the image brighter. How the gamma correction works is described in Subsection 3.2.3.

3.3.4 Data Selection based on U-net Performance

Before the data is used for training, we want to validate the quality of the dataset. Each candidate dataset is used to train a separate U-net model [18] integrated in the pipeline described in Subsection 3.3.3, replacing the transformer denoising. The real sensor data has no ground truth pairs. Therefore, an unseen test set is created by randomly selecting 10% of each public avail-

able dataset described in 3.3.1. The test set images will be centre cropped to 448×448 because of computational reasons. This test set is used for generating quantitative results. In addition, the separate trained models will be tested on the real sensor data, which generates qualitative results.

For visualising the feature differences between the datasets, we first extract the features using the VGG16 model [96]. The VGG16 model is a widely used model for extracting features from images [97]. The feature output is $(n * 7 * 7 * 512)$ which makes it impossible to visualise. We use t-SNE [85] to reduce the feature space to two dimensions, which enables the visualisation of the features.

To reduce the training time even more, we also want to test the performance drop by using a sample of the selected datasets. The expectation is that denoising performance will decrease by sampling the data. However, if this would not be the case, it saves training time. For the samples, we randomly select a half and a quarter of all the datasets.

3.4 Teacher Model for Raw Denoising

This subsection describes the methodology used for the teacher network. Two transformer models are used to determine the best teacher model, being Restormer [65] and SCUNet [68]. Both Restormer and SCUNet are originally used to denoise sRGB images (3 channel input), while we aim to denoise raw images (4 channel input). Raw denoising requires an integrated upsampling step in the model [98]. Therefore, some changes will be made to both models. The network structure of Restormer is discussed in Subsection 3.4.1 and the network structure of SCUNet is described in Subsection 3.4.2.

3.4.1 Modified Restormer Architecture

The original Restormer architecture for denoising sRGB images is described in Subsection 2.2.2. The modified Restormer model is shown in 3.13. The skip connection from the input image to the restored image (the upper above

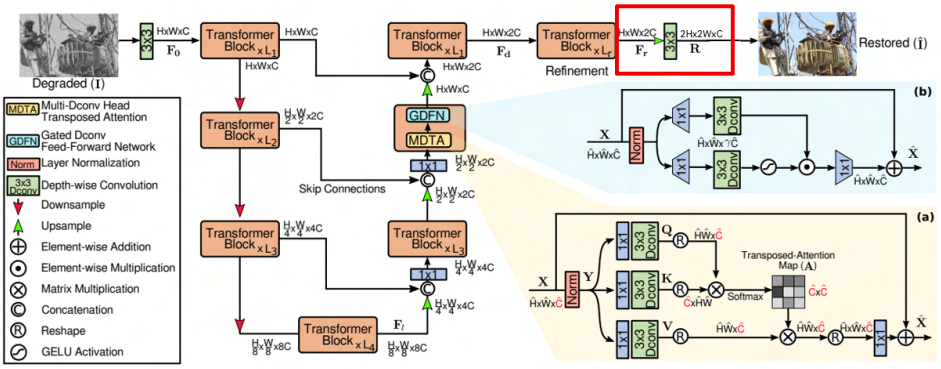


Figure 3.13: Modified Restormer Model

one in Figure 2.5) is not used for real noise denoising in the original proposal because of performance reasons [65], which is why we will also not use it.

The number of transformer blocks is also something to determine. Using more or less transformer blocks at the resolution steps is a method to scale the Restormer model. The number of blocks used for the teacher model are respectively 4, 6, 6, 8 for the parameters L1, L2, L3 and L4. The number of heads for each block is also scalable. We use 2 heads at each block. These settings are based on the original proposed model for real noise image denoising [65].

To enable raw to sRGB denoising, an upsampling method is added between the last transformer block and the last 3×3 convolution layer. If we would upsample at the beginning, all feature map shapes would be doubled. Bigger feature maps result in a longer prediction time. Therefore, we will place the upsample step at the end of the model.

We decided to apply a transposed convolution operator because of the good performance [99]. A transposed convolution task is the reverse operation of a regular convolution task and is commonly used for image generation tasks where a higher resolution is desired [65], [68].

3.4.2 Modified SCUNet Architecture

The modified SCUNet is shown in 3.14. The goal of the SCUNet is to denoise sRGB images. Therefore, we add an extra upsampling between the last SC Block and the last 3x3 convolution layer, just like we did at the Restormer

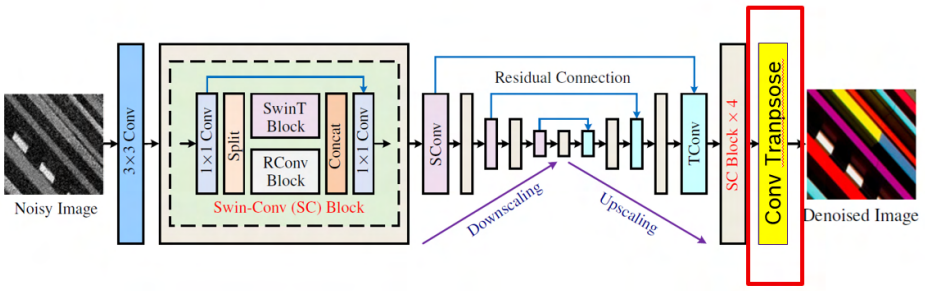


Figure 3.14: Modified SCUNet Model

model. Besides adding an extra upsampling for raw denoising, we also changed the padding from replication padding to reflection padding. The original padding caused a border problem in our data (See Figure 3.15a). The border problem only appeared when tested on data which was not dividable by 64.

(a) Border problem **(b)** Apply reflection padding

Figure 3.15: Border problem for SCUNet default training strategy

We tried multiple padding options. When applying reflection padding instead of replication padding, the border artefacts are gone. The result for using reflection padding is shown in 3.15b.

3.4.3 Training Strategy for the Teacher Network

The first phase for knowledge distillation is to train the teacher. We trained two teacher models, one based on the Restormer [65] model and one based on the SCUNet [68]. The training strategy for both models started based

on their original proposal, but after some experiments the training strategy changed. More about those experiments is described in Chapter 4. The final training strategy is described further in this section. First, we describe how we initialise the weights in Subsection 3.4.3.1. Then we describe in Subsection 3.4.3.2 about progressive learning for Restormer. At last, the teacher loss is described in Subsection 3.4.3.3.

3.4.3.1 Initialising the Weights

The first step before the training starts is to initialise the weights of a model. For both the Restormer and SCUNet based models, the weights are initialised by copying the weights from the pre-trained models. Initialising the weights from a model with a similar goal is beneficial for the training speed because the weights have an indication which is probably closer to the convergence point than with random weights. The adjustment for both models from a three channel input to a four channel input precludes the initialisation for the input weights from the original models. The extra upsampling step is also not trained in the original proposal. Therefore, these weights are initialised by a random value between 0 and 1.

3.4.3.2 Progressive Learning

The original Restormer weights are trained using progressive learning. Progressive learning is a method to increase the patch size and decrease the batch size multiple times after completing a certain number of iterations. The goal for progressive learning is for the transformer model to learn features at different levels of detail. W found that the progressive learning performs significantly better for the Restormer model, which is why we decided to use it for training the Restormer-based model. Because of the significant improvement in performance, we also tried applying progressive learning to the SCUNet based model. Despite the improvement for the Restormer model, using progressive learning did not improve the results for the SCUNet compared to the original strategy with a fixed patch and batch size. This could be because SCUNet already has a combination of a transformer block and a CNN block for extracting non-local and local features,

while Restormer only uses transformer blocks.

3.4.3.3 Teacher Loss

The teacher loss is based on the loss used to train the original SCUNet [68], [100]. The loss is a combination of a pixel loss, a perceptual loss and an adversarial loss. Pixel loss is a pixel wise comparison between the prediction and the reference image. As pixel loss, we use L1 loss, defined by:

$$l_{pixel} = \sum_{i=1}^{n} |y_i - \hat{y}_i|$$

Where n is the number of pixels, y_i represents the reference pixels and $\hat{y}_i$ represents the predicted pixels.

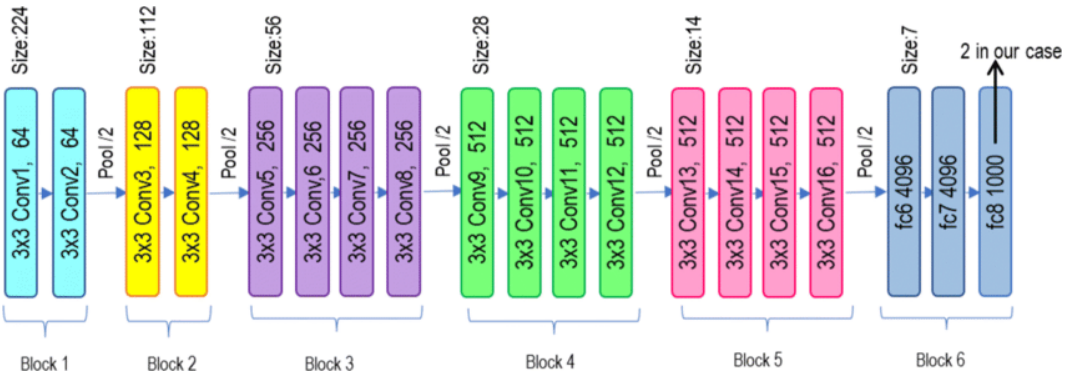
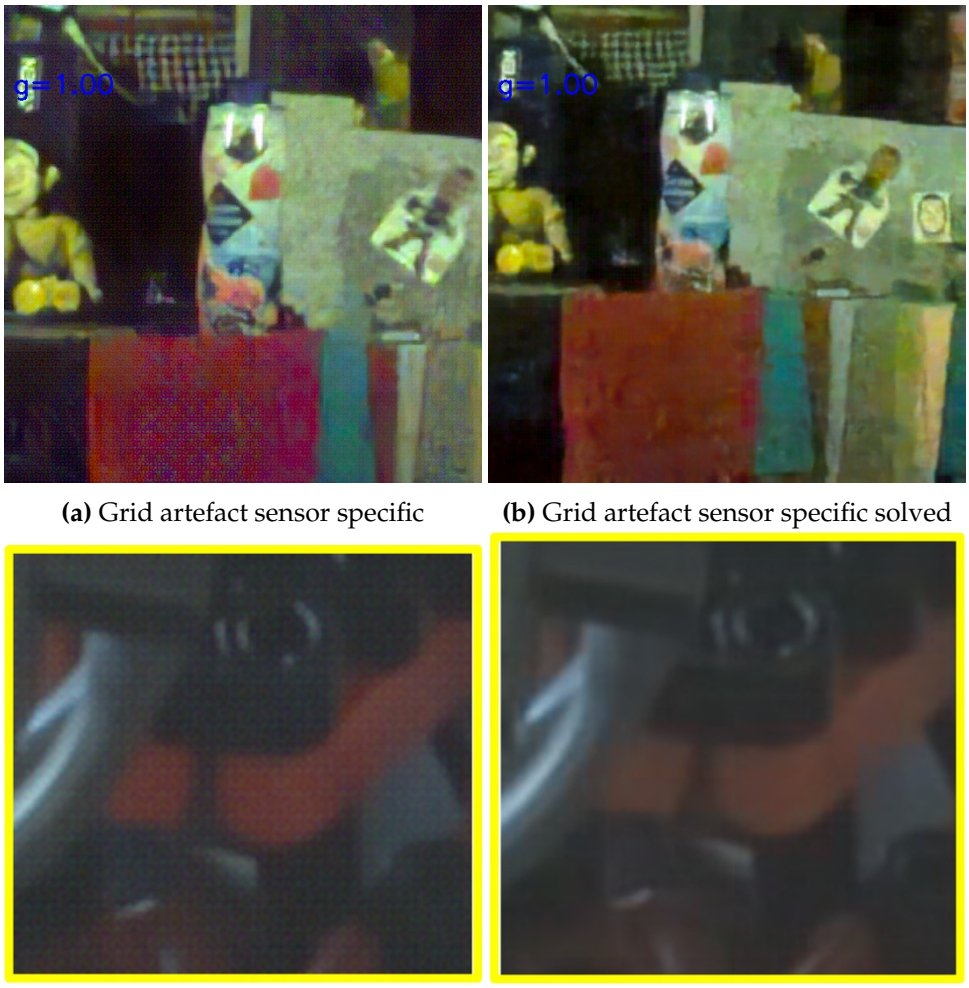


Figure 3.16: VGG-19 architecture [101]

In addition to the pixel loss, we use the perceptual loss. Perceptual loss is used to match the features between the reference data and the prediction. For the feature extraction, we use the pre-trained VGG-19 model [102] because VGG-19 is known to be a good feature extractor and VGG-19 is used in the SCUNet proposal [68], [103]. After the features are extracted for both the predicted and reference image, L1 loss is used to calculate the differences between them. The SCUNet proposal uses the layers Conv1_2, Conv2_2, Conv3_4, Conv4_4 and Conv5_4 (see Figure 3.16). If we use the same layers for the perceptual loss, a grid artifact appears in the results (see Figure 3.17a).



(a) Grid artefact sensor specific **(b)** Grid artefact sensor specific solved

(c) Grid artefact public unified **(d)** Grid artefact public unified solved

Figure 3.17: Grid artefacts for SCUNet default perceptual loss

Perceptual loss can cause these kinds of grid artefacts [104]. We decided to train a SCUNet model without the perceptual loss to see if the perceptual loss is causing the pattern. For the model, we use the same training strategy used for the baseline model, except for removing the perceptual loss.

Loss function	PSNR ↑	SSIM ↑
With L_{per}	37.150	0.906
No L_{per}	36.537	0.899

Table 3.3: Dataset details based on 1950 centre cropped 448×448 images of the unified test, with 60 dB of noise.

When we remove the perceptual loss, we see that the grid artefacts disappear (see 3.17). What we also see is that the performance on details is worse. When we look at Table 3.3, we see that the performance on the uni-

fied dataset also significantly decreased. The perceptual loss used to train SCUNet consists of five different layers, being Conv1_2, Conv2_2, Conv3_4, Conv4_4 and Conv5_4. These are the last layers from the first five blocks of the VGG19 layers. After these layers, ReLu is applied to the features. It is more usual to extract the features from the ReLu layers [104]. Therefore, we first try to use ReLu1_2, ReLu2_2, ReLu3_4, ReLu4_4 and ReLu5_4 layers for the perceptual loss. We train a model using each separate layer as the perceptual loss, to see which layer causes the grid artefacts. We keep the set weights from the SCUNet training (0.1, 0.1, 1, 1, 1) for the layers. The results for these models are shown in Appendix B. From the results, we see that the border effect is only visible for the models trained with the ReLu3_4 and ReLu4_4 layers. We therefore choose to continue the training only using ReLu1_2, ReLu2_2 and ReLu5_4. This results in the following loss function:

$$L_{per} = 0.1 * L_1(ReLu1\_2(y), ReLu1\_2(\hat{y})) + 0.1 * L_1(ReLu2\_2(y), ReLu2\_2(\hat{y})) + 1 * L_1(ReLu5\_4(y), ReLu5\_4(\hat{y}))$$

The last addition to the teacher loss is the adversarial loss. The adversarial loss is used to create and control the competition between the encoder and the decoder. For the adversarial loss, we use a combination of the sigmoid and BCE loss [105]. The adversarial loss is defined by:

$$l_{adv} = \frac{\sum_{i=1}^{n} -w_i[y_i * \log \sigma(\hat{y}_i) + (1 - y_i) * \log(1 - \sigma(x_i))]}{n}$$

Where n is the batch size, $\hat{y}_i$ is the prediction, y_i is 0 or 1 if the generated image is real and w_i is an optional weight for each sample which we do not use.

These loss functions together form the teacher loss, which can be defined

as:

$$l_{teacher} = \alpha l_{pixel} + \beta l_{per} + \gamma l_{adv}$$

For training, we use $\alpha = 10$, $\beta = 1$ and $\gamma = 1$. In the SCUNet training proposal they use $\alpha = 1$, $\beta = 1$ and $\gamma = 1$ but we saw that the pixels loss started and stayed significantly lower during training than the other losses, which would result in little influence. Therefore, we chose to set α to 10.

The proposed training strategy for Restormer suggests to only use l_1 loss. However, in the experiments we saw that using only l_1 loss performed bad for restoring colors, which is why we decided to use the $l_{teacher}$ also for the Restormer model. The experiment is shown in Section 4.2.1.

3.5 Distilling Knowledge to the Student

After the training of the teacher model is done, the student model can be trained. As described in Section 2.3, The goal while training the student model is to transfer the knowledge from the teacher model to the student model. To achieve the knowledge transfer, we use some loss functions based on the teacher predictions. In Section 2.3 we saw that response-based loss and feature-based loss perform best for image denoising. For response-based loss, we use L1 loss between the student prediction and the teacher prediction, resulting in:

$$l_{response} = \sum_{i=1}^{n} |\hat{y}_{i,teacher} - \hat{y}_{i,student}|$$

For the feature loss, we use the feature maps at the end of each resolution step. Between the feature maps, we also use the L1 loss to calculate the difference, resulting in:

$$l_{feature} = \sum_{j=1}^{m} w_j \sum_{i=1}^{n} |f_{ij,teacher} - f_{ij,student}|$$

Where m is the number of resolution steps, w_j is a weight to set the trade of between the different feature maps, and f_{ij} is the feature value i from feature map j.

To form the total student loss, we add the teacher loss functions to the knowledge distillation losses, resulting in the following formulas:

$l_{student\_response} = \alpha l_{pixel} + \beta l_{per} + \gamma l_{adv} + \delta l_{response}$

$l_{student\_feature} = \alpha l_{pixel} + \beta l_{per} + \gamma l_{adv} + \epsilon l_{feature}$

From Section 2.3.3 we have learned that for using knowledge distillation for transformer models, it is better for the accuracy of the model to reduce the number of blocks over the number of channels. We therefore will compress the teacher model blocks to halve the number of blocks and a quarter of the number of blocks to examine the knowledge distillation effectiveness.

4. Experiments and Results

This chapter describes the experiments and results of training the denoising transformer using knowledge distillation. As described in Section 3.1, our research consists of three stages. For the first phase, a selection of the unified data will be examined in Section 4.1. The goal of the second phase is to optimize the performance of the teacher model. The experiments for optimizing the teacher network are described in Section 4.2. The teacher model is then used to transfer the knowledge to a computational less heavy student model. The experiments about distilling the knowledge from the teacher to the student are described in Section 4.3.

For testing the performance of the trained models on the unified dataset, we use two metrics, peak signal-to-noise (PSNR) and structure similarity index (SSIM). PSNR and SSIM are image quality metrics. PSNR is based on the mean squared error (MSE). When MSE approaches zero, PSNR approaches infinity [106]. SSIM is an image comparison method considered to be correlated with the human visual system. SSIM uses three loss-based factors to calculate the similarity, instead of using only one [106]. SSIM tends to be

4.1 Dataset Selection

This section describes the results of the data selection experiment. First, the quantitative results are shown in 4.1.1. Then the qualitative results are shown in 4.1.1. Next, the feature differences are displayed using t-SNE in 4.1.3. At last, the sampling of the data is described in 4.1.4.

4.1.1 Quantitative Data Selection Results

This subsection will discuss the results for the data selection. The models, trained on the separate datasets, are tested on the target data with a noise level of 60db. 60db is a high level of noise which is typically only occurs in low-light conditions. The input image with 60db is shown in Figure A.1b. Table 4.1 the quantitative values are shown for each separate trained model.

Dataset	PSNR ↑	SSIM ↑
HDR+	32.413	0.883
SIDD	31.993	0.865
SID	31.240	0.868
Raise	**34.965**	**0.891**
Flickr2K	34.135	0.886
Mit5k	34.417	0.885
All combined	34.637	**0.893**
Selected	**34.794**	0.890

Table 4.1: Dataset selection quantitative results on 60db noise

In Table 4.1 the quantitative values are shown for each separate trained model. The model trained on all data is the baseline for the performance comparison of the models. Table 4.1 shows that the models trained on HDR+, SIDD and SID perform significantly worse on PSNR than the model trained on all the data. On SSIM the HDR+ scores similar to the baseline model, but SIDD and SID still perform significantly worse. Raise is the best performing separate model, scoring even better on the PSNR than the model trained on all data. This implies that some data decreases the performance of the model. Mit5k scores on PSNR and SSIM slightly worse than the model trained on all dataset. Flickr2k performs on SSIM similar to the baseline model, but on PSNR it performs worse.

Overall, we see that it is important to make a selection from the datasets because Raise on its own performs better than the model trained on all datasets. Further, we see that SIDD and SID perform significantly less on PSNR and SSIM than the model trained on all data.

4.1.2 Qualitative Data Selection results

For the qualitative comparison, we use the same models used in Section4.1.1. To get an insight in the visual performance, we show single frame results of the separate models in Figure A.1.

If we compare the results of the separate trained models against the model trained on all the datasets, we can see that the model trained on HDR+ removes the noise comparable with the model trained on all the datasets. Especially in the dark regions, the model trained on HDR+ seems to clear the noise well compared to the model trained on all data. The model trained on SIDD has little fade compared to the model trained on all data. We use the term fade to describe the circular faint spots as in shown in A.1g. Despite the little fade, the image is more blurry than the image from the model trained on all data. The image produced by the model trained on the SID dataset has also little fade, but the level of detail is also low. The model trained on Raise performs similar to the model trained on all data, with a little more blur. The model trained on Flickr2k generates more fade, mainly in the dark parts. Despite the fade, the image has a good amount of detail. At last, we compare the model trained on Mit5k. The image generated by the model performs similar to the model trained on HDR+ except for the fade. The Mit5k model generates more fade than the HDR+ model.

When we compare these visual results to the quantitative results from 4.1.1, we see that the low PSNR and SSIM from SIDD and SID are confirmed by the low performance on the visible results. In the case of HDR+, we observe that the visible outcomes align more closely with the similar SSIM as the model trained on all, rather than the lower PSNR. For Flickr2K we see the opposite. Based on the qualitative results, you would expect Flickr2K to perform better on the visual results.

Practical applications often use the approach to merge multiple frames for a better denoising performance [24]. They take the average over multiple frames, resulting in less noise. Therefore, we also show the averaged results of the denoising models in Figure A.2. The frames are averaged based on 100 samples with the noise level of 60db.

The averaged frame results in Figure A.2 a better visual of the quality of the reconstruction of the image. It is easier to see the generated colors and details of the images. The averaged results confirm the previous claim that the colors are off for the model trained on the SID dataset, having a greyish look. The colors of the SIDD model look more realistic than the other models. If we look at the details, we see that the SIDD model still lacks of detail compared to the other models. The Raise model has slightly more detail than the other models.

Based on the qualitative and quantitative performances, we chose to use the HDR+, Raise and Mit5k. Raise and Mit5k score high on both qualitative metrics and on the single and average target data comparison, which make them a clear choice to add in the selected dataset. HDR+ scores bad on the PSNR, but it scores high on SSIM and on the regenerated target data, which is more important for the research. Flickr2k scores high on the unified test set, but the lack of performance on the target data is too significant to include them in the selected dataset. The SIDD and SID dataset score bad on both the quantitative metrics of the unified test set and the visual results of the target test set.

When we compare the selected dataset from Table 4.1 we see that based on PSNR it has a higher performance than the model trained on all the data. Based on SSIM it performs comparable to the model trained on all data. The visual results confirm these statistics. The model trained on the selected data contains a bit less fade and has more detail.

4.1.3 The Differences Between the Datasets Using t-SNE

We use a t-SNE plot to highlight the similarities in the datasets. To generate this plot, we first extract features of the datasets with the pre-trained VGG16 model [96]. For predicting the features, we leave the fully connected layers out of the model to extract the features of the images. When we get the features, we flatten them and use t-SNE to reduce the dimensions to two. The t-SNE visualisation for all data is shown in Figure 4.1.

In Figure 4.1 we see a strong group of mostly purple (Raise) at the left.

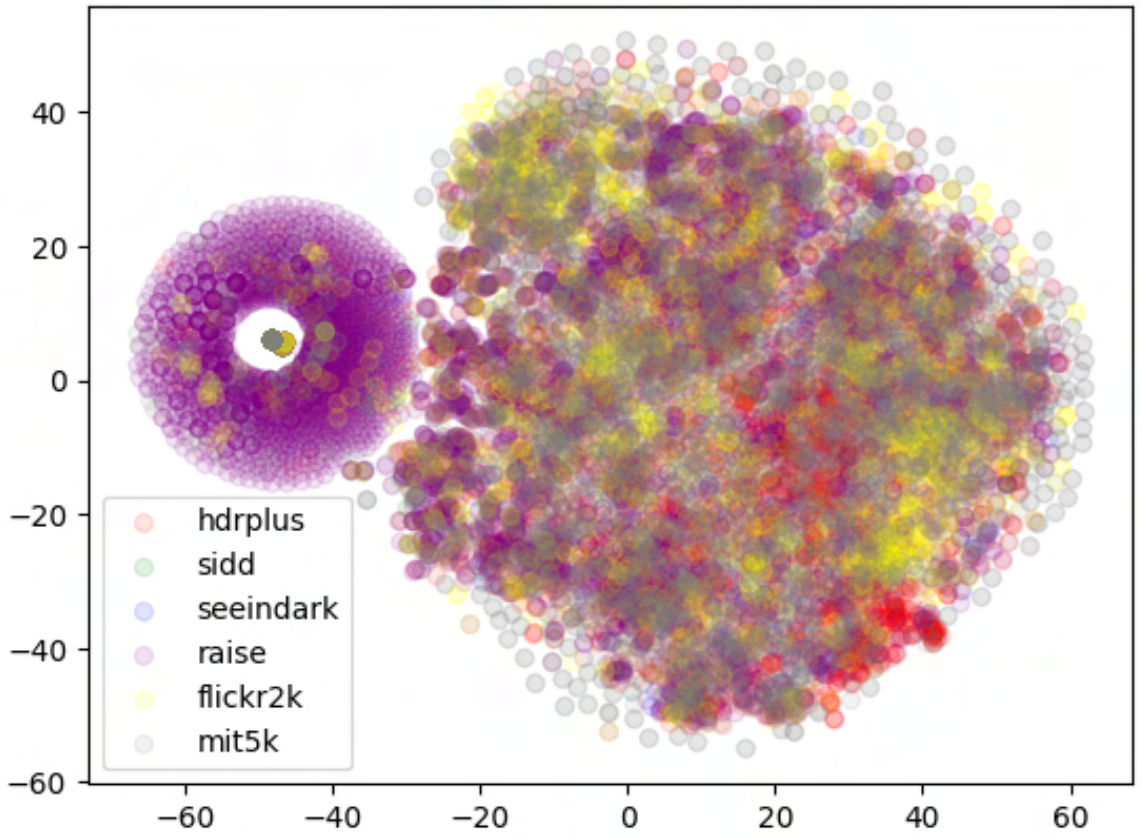


Figure 4.1: t-SNE highlighting the feature differences between the potential data sets.

The rest of the data seems more evenly spread in a bigger cloud. This implicates that Raise has different features in within the data than the other datasets. When we dive deeper in the features of Raise [91] we see that unlike the other dataset, Raise is not focussed on low light images. The rest of the features are similar to the other datasets, which implies that the big feature difference cluster is based on the high light image.

Based on the quantitative (Section 4.1.1) and qualitative results (Section 4.1.2) we decided to continue with the datasets HDR+, Raise and Mit5k. In the t-SNE from Figure 4.1 we see that the yellow dots from the flickr2k dataset are slightly spread over the cloud on the right. The red dots are more available in the right bottom corner of the t-SNE plot. The green and blue dots, from the SIDD and the SID dataset are more even spread over the figure. This implies that the SIDD and SID dataset are more similar to the other datasets. The reason for their less performance is probably that they

have a few samples, making it harder for a model to learn the patterns of the data. The Flickr2k dataset is the only dataset based on sRGB images. The results implicate that the unprocessed sRGB images are not as good for the raw denoising as the original raw images.

4.1.4 Data Reduction

We also want to test For testing the performance drop when using a smaller dataset, the same model is trained on half of the samples and a quarter of the samples from the combined dataset. These smaller datasets are created by sampling for each dataset separately, making sure the ratio between datasets remains. The quantitative results for the sampled datasets are shown in Table 4.2

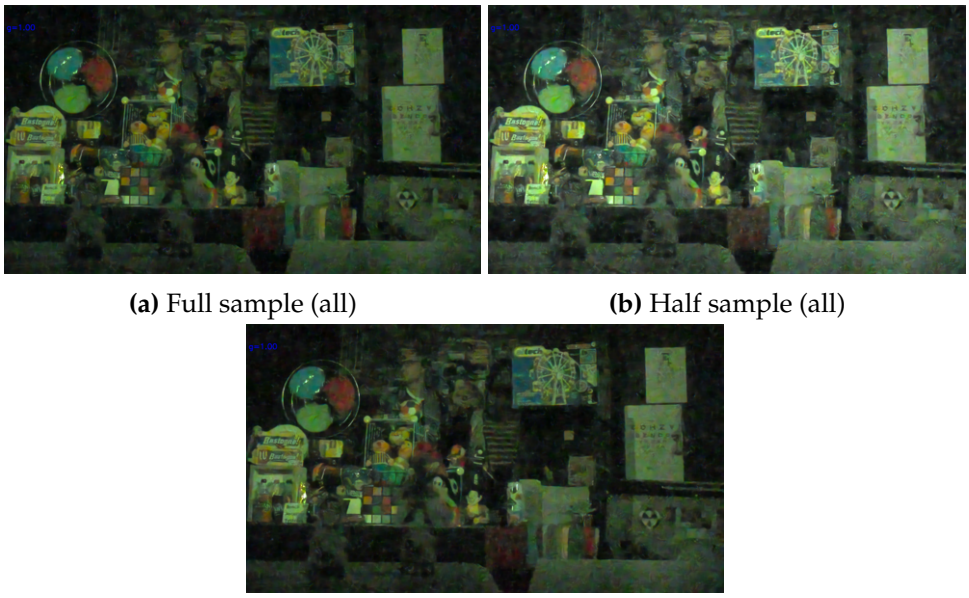
Dataset	PSNR ↑	SSIM ↑
Full sample (all)	34.637	0.893
Half sample (all)	34.738	0.891
Quarter sample (all)	34.932	0.891

Table 4.2: Data sample selection quantitative results on 60db noise

The results in Table 4.2 show that the PSNR of the half and quarter sample is slightly higher than the PSNR from the model trained on all the data. This is a noticeable result because the performance of a model is expected to go up if there is more data available for training. The SSIM shows a slight decrease in performance when the data is reduced, which is more expected. The results imply that the performance would not decrease significantly when using a quarter of the available dataset. However, the differences are small, which would indicate that it would be beneficial to only use a sample of the dataset.

For testing the performance drop when using a smaller dataset, the same test set is used for the models trained on half and a quarter of the data. The single frame results are shown in Figure 4.2.

The single frame results in Figure 4.2 show that the amount of detail decreases when we halve the training data. The quarter sampled data has more detail than the half sampled data. However, the fade increases signifi-

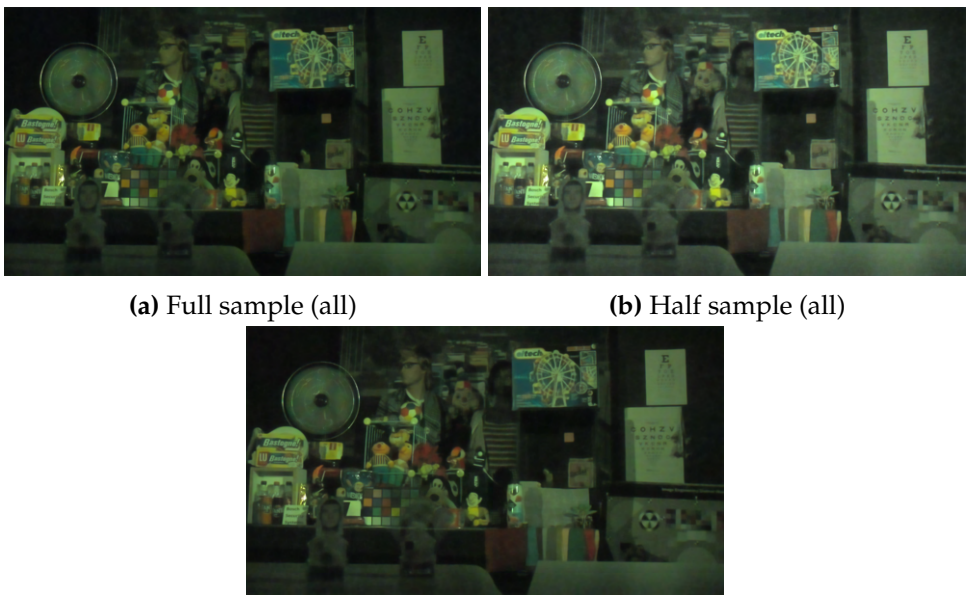


(a) Full sample (all) **(b)** Half sample (all)

(c) Quarter sample (all)

Figure 4.2: U-net trained on separate samples of the combined test set, tested on target test set with single frames containing 60 db noise.

cantly each time we halve the dataset, which implies that it would decrease the performance of the model if a smaller sample of the available data is used.



(a) Full sample (all) **(b)** Half sample (all)

(c) Quarter sample (all)

Figure 4.3: U-net trained on separate samples of the combined test set, tested on target test set with averaged images containing 60 db noise.

The averaged results of 100 frames with 60db of noise are shown in Table

4.3 to have a more clear sight in the details and color reconstruction. They show that the amount of detail decreases in each step of halving the dataset. In addition to the decreasing amount of detail are the colors of the generated images from the smaller datasets more greyish than the image generated by the model trained on all data.

Based on the decreasing detail in both the single frame result and the averaged frame results, we decided that the performance drop of the model would suffer too much for sampling the selected data.

4.2 Teacher Network Optimization

For the experiments, we decided to use the proposed real noise training strategies from SCUNet [68] and Restormer [65] (see Section 3.4). SCUNet uses a patch size of 544×544 with a batch size of 24. We reduced the patch size to 224×224 and the batch size to 16 because of computational limitations. A step-wise learning rate is used starting from 0.0001. The optimizer is Adam [107].

For the Restormer the Adam optimizer is also used with a linear learning rate is used starting at 0.0001. The proposed progressive learning for the Restormer model is with patch and batch sizes of [(128x128, 64), (160x160, 40), (192x192, 32), (256x256, 16), (320x320, 8), (384x384, 8)] updated at iterations $[92K, 156K, 204K, 240K, 276K]$ [65]. For computational reasons, we reduce the batch size with a factor four, resulting in [(128x128, 16), (160x160, 10), (192x192, 8), (256x256, 4), (320x320, 2), (384x384, 2)].

4.2.1 Restormer with SCU loss

The results from the baseline Restormer model contain little color, and they still contain noise (see Figure C.1, C.7 and C.13, containing 32db, 45db and 63db of noise). The reason could be that the L1 loss is based on the differences between the pixel values, without taking important structures within the pixels into account [108], [109]. Therefore, we chose to train a new Restormer model with the same loss as the SCUNet model. Table 4.3 shows

the qualitative results for the experiment.

Loss function	PSNR ↑	SSIM ↑
$L1$	34.392	0.874
$l_{teacher}$ loss	37.459	0.911

Table 4.3: Restormer performance based on 1950 center cropped 448×448 images of the unified test, with 60 dB of noise.

In Table 4.3 we see that based on both PSNR and SSIM the performance of the Restormer model has increased significantly. If we look at the visual results in Figures C.1, C.7 and C.13, containing 32db, 45db and 63db of noise, we see that the colors are restored, and the noise is significantly less. From this experiment, we can say that the SCUNet-based loss works better for the Restormer on our target data. Combining the L1 loss with adversarial loss and perceptual loss allows for a more comprehensive and robust optimization process. Each loss function captures different aspects of the image reconstruction task. By combining them, the model can learn to balance various trade-offs and produce higher-quality results

4.2.2 Best Performing Teacher Model

Before knowledge distillation is applied, we want to compare which teacher model is performing best. To validate that the transformers perform better for the real sensor noise, we decided to also compare the U-net trained on the selected dataset. The quantitative results are shown in Table 4.4.

Modified Architecture	PSNR ↑	SSIM ↑	parameters (M) ↓	GFlops ↓
SCUNet	37.254	0.916	17.963	272
Restormer	38.536	0.931	26.154	463
U-net	35.022	0.903	0.766	77.2

Table 4.4: Deep denoising performance based on 1950 center cropped 448×448 images of the unified test, with 60 dB of noise.

In Table 4.4 we see that the transformer networks outperform U-net significantly. We also see that Restormer performs better than the SCUNet based on PSNR. If we look at the SSIM, the models are more comparable. The qualitative results are shown in Appendix D. If we look at the qualitative real sensor results, we see that for small noise (32db) the differences are

small, but if we look at the higher noise level (63db) the differences become more visual. Especially in terms of color the transformer models perform better than U-net. They also contain a bit more detail. If we look at the differences between SCUNet and Restormer we see that Restormer has slightly better details but SCUNet has less remaining noise. PSNR and SSIM form the good bases for the image comparison. PSNR has a weakness for measuring fidelity signal, resulting in a low representation for the perceptual quality. SSIM represents the perceptual quality better, which is why SSIM is more important for our comparison [108], [110]. For SSIM the differences in performance are small, which means that the visual results are required for making a good dicision. For the visual results, the SCUNet has less detail but also less remaining noise. The focus task is denoising, which is why we chose to continue with SCUNet for knowledge distillation.

4.3 Student Network Learning via Knowledge Distillation

For the student model, we want to compress the SCUNet-based in multiple steps to see the effect for each compression level. Therefore, one student model contains half the number of blocks and the second student model contains a quarter of the blocks. We use the same training strategy as for the teacher model, except for the student loss. We also have to decrease the batch size to 12 because we extract the features while training.

For the knowledge distillation, we want to compare the results from the methods against the teacher model and the student model trained from scratch. The comparison with the teacher model will show how much performance there is lost with compiling the model. The comparison with the student model trained from scratch will show how much influence the knowledge distillation has had on the performance of the student model. The results of the experiment are shown in Table 4.5.

If we look at Table 4.5 the surprising thing is that the PSNR of the SS2 model is higher than the teacher model. This is surprising because we

ID	Knowledge distillation	Blocks	PSNR ↑	SSIM ↑	parameters (M) ↓	GFlops ↓
T	Teacher model	4	37.254	0.916	18.0	272
SS2	Student model from scratch	2	37.524	0.920	9.7	163
SS1	Student model from scratch	1	37.246	0.917	5.5	108
SR2	Response based	2	37.290	0.916	9.7	163
SF2	Feature based	2	37.561	0.921	9.7	163
SR1	Response based	1	37.405	0.919	5.5	108
SF1	Feature based	1	37.363	0.918	5.5	108

Table 4.5: Knowledge distillation results based on 1950 center cropped 448×448 images of the unified test, with 60 dB of noise.

would expect to lose performance on accuracy when compressing a model. The SSIM of the student model from scratch is comparable with the teacher model because the difference is small. For SS1 we see that the student model trained from scratch performs slightly lower on both PSNR and SSIM, which is more in line with the expectations.

If we look at the PSNR for the SF2 model, we see that it is slightly higher than the teacher model and comparable with SS2. The SSIM for the feature-based model is comparable with the teacher model and the SS2. The SR2 performs similar to the teacher model on both PSNR and SSIM. Because of the small differences in the quantitative performance, it is important to look at the visual results for the performance comparison.

The qualitative data is shown in Appendix E. If we look at the visual results for the teacher model and SS2, we see that they make more sense. The teacher model has more detail and less fade than the SS2 and the SS1 model. The response-based models perform significantly worse than the

teacher model, but also significantly better than the student model trained from scratch. The feature-based models perform slightly worse than the teacher model, but they perform significantly better than the student model trained from scratch. The differences are the best visible in the higher noise images.

The qualitative results from the unified dataset implicate that the knowledge distillation has no positive effect on the model performance of the student model, and that SS2 and SS1 perform comparable to the teacher network. However, the qualitative results from the real sensor data show a significant improvement using knowledge distillation when compressing the models to half the blocks, especially for the feature-based model. From these statements, we can learn that it is beneficial to use knowledge distillation to improve small model performance. We also see that little accuracy is lost, and much efficiency is gained by compressing a bigger model using knowledge distillation.

5. Conclusion

The goal of the research is to find the right balance between efficiency and accuracy for real sensor denoising. We have gone through three phases before cumming to the final denoising model. First, we unified and selected a raw dataset. The research question applicable for this stage was:

2. How to generate pair-wise realistic noisy data on a large scale?

To answer this question, we answer the specified sub-questions. The first sub-question was:

2a. How to unify publicly available raw denoising datasets for the use of training a deep denoising model?

For this research question, we proposed a unification pipeline. We unified multiple public available raw datasets using the unification pipeline. The processed were used as clean images. We used a real-world noise model to generate the noisy pairs. The quality of the processed images were tested using U-net model. We found that the processing pipeline was resulting in good denoising results for the real sensor images. The next sub-question to answer was

2b. What is the impact of the white balance and color correction matrix approximation for converting sRGB images back to raw images?

To answer this sub-question, we used an additional unprocessing pipeline to the unification pipeline. This unprocessing pipeline estimates the white balance and color correction matrix. For the unprocessed unified images, we used the same test as the unified raw images. Based on this test, we found that the white balance and color correction matrix approximation were not good enough to include the Flickr2k sRGB dataset in the selected dataset. The last sub-question in this phase was:

How large should the training dataset be for training the teacher network?

We trained a U-net model on samples of the dataset, testing how important the amount of data was for the quality of the prediction. From this test, we found that using a smaller sample of the selected dataset would harm the accuracy too much.

The next phase was to train the teacher network. For this stage, there were no research questions to answer. However, this stage was required to enable the knowledge distillation. For training the teacher network, we optimised the Restormer and SCUNet model. In addition to the original networks, we added an upsampling layer to the Restormer and SCUNet to make raw denoising possible. During the optimization, we resolved a border problem caused by replication padding and a grid artifact caused by specific layers of the perceptual loss. We also found that the $L1$ loss for training the Restormer was not good for our real sensor data, which is why we added the perceptual loss and adversarial loss. The transformer models outperformed the U-Net on the unified testset and on the real sensor data. The differences in performance were more clear by a higher noise level. For the transformer comparison, SCUNet turned out to be more applicable based on a comparable performance with a more efficient model. This resulted in SCUNet forming a basis for the knowledge distillation.

In the last stage, we compressed the SCUNet into a more efficient student model by using response-based and feature-based knowledge distillation. In this stage, we tried to answer the following research question:

1. What is the impact of compressing a transformer model for real world low-light image denoising using knowledge distillation?

To answer this question, we answer the specified sub-questions. The first sub-question was:

1a. How much knowledge is lost after compressing the model using knowledge distillation?

To answer this sub-question, we used response-based and feature-based knowledge distillation. The student model trained with response-based knowledge distillation had a significant drop in performance. The student

model trained with feature-based knowledge distillation only had a small performance drop.

1b. How much computing power is saved after compressing the model using knowledge distillation?

For the student models, we compressed the teacher model twice. We used half of the blocks and a quarter of the blocks. Half the blocks gains an efficiency of approximately 45% and a quarter of the blocks gains an efficiency of approximately 70%.

1c. What is the performance difference of the student network compared to the same network structure being trained from scratch?

Compressing the model without using knowledge distillation causes a significant performance drop. Using response-based knowledge distillation slightly improves the student model's performance. For a better result, you can apply feature-based knowledge distillation because that method significantly improved the performance on the real sensor data.

5.1 Discussion and Future Work

This section describes the discussion and future work points of the research. First, the conflicting results between the quantitative and qualitative results are described in Subsection 5.1.1. Then the batch reduction of the progressive learning strategy from the Restormer model is discussed in Subsection 5.1.2. At last, some techniques to finalise the student model performance are described in Subsection 5.1.3.

5.1.1 Conflicting Results Between Quantitative Results and Qualitative Results

Throughout the experiments, we have observed that the quantitative metrics implied the opposite as the qualitative results. This was mainly true for PSNR. Calculating the PSNR is based on pixel values, which results that the perceptual differences and structural information is not considered when

calculating the loss. This could lead to different PSNR quality values than the quality differences shown in an image [110]. There is some work which aims to improve the metrics over PSNR. However, the possible improvements are too complex to use, which is why PSNR is currently still widely used [110].

SSIM showed less difference in the experiments and was more in line with the qualitative results. SSIM is calculated based on luminance, contrast, and structural similarity, representing the perceptual quality better. This is why SSIM is a better quality metrics for the reconstruction of the image than PSNR [108], [110]. There is numerous research which tries to improve the metrics for image quality comparison. However, it tends to be challenging to get all the quality measures into a number [110].

5.1.2 Progressive Learning Strategy

For training the Restormer we used the proposed progressive learning strategy [65]. Because of computational reasons, we could not follow the proposed batch size for the strategy. Finding the optimal progressive learning for the best performance is about finding the right trade-off iterations for changing the patch- and batch sizes. When the batch sizes are decreased, it could result in different trade off between the other progressive learning parameters [111]. We reduced the batch size by a factor of four, but we did not optimise the number of iterations for changing the batch and patch size. For future work, the progressive learning strategy could be optimised for the smaller batch size. More available computing power could also enable the ability to train with the original proposed progressive learning strategy.

5.1.3 Student Model Optimizing

When training the student model, we used real-time predictions from the teacher model to generate the teacher loss. Because of limited computing power, we had to reduce the batch size for training the student model. Reducing the batch size can result in a lower accuracy, so for optimizing the student model, the same batch size as the training could be beneficial for

the model accuracy. For future work, the teacher features and results could be saved, which would save computing power, so the student model could be trained on the same batch size as the teacher network. This saves a lot of computational power for the teacher network, but it will take a bit of (low-cost) memory.

Further, the student model is only trained on unified publicly available data with a noise model to generate the image pairs. Using the real-world noise model is close to the real noise images. However, we only add noise to the images. The other features which correlate with noise, like image darkness, are not simulated. Therefore, we expect that training the model first on the generated simulated data, and then fine tune on the real noise data is expected to improve the model accuracy [112].

For applying the knowledge distillation, we only applied response-based and feature-based knowledge distillation separately. In Subsection 2.3 we saw that knowledge distillation techniques are often combined. To improve the effect of knowledge distillation, the feature-based and response-based methods could be combined. We also set the loss trade-off weights for the distillation loss to one. For further improvement of the knowledge distillation effect, the trade-off values for the loss weights could be tweaked. Because we did not use multiple weighs, we are not sure that the weights are optimal. We would expect that the weight of the feature-based knowledge distillation should be higher because the performance went significantly up when we added the feature distillation loss.

Bibliography

[1] M. Brown and S. Kim, "Understanding color and the in-camera image processing pipeline for computer vision," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Tutorial*, 2019, pp. 1–247.

[2] D. Liu, B. Wen, X. Liu, Z. Wang, and T. S. Huang, "When image denoising meets high-level vision tasks: A deep learning approach," *arXiv preprint arXiv:1706.04284*, 2017.

[3] H. Wang, Y. Li, Y. Wang, H. Hu, and M.-H. Yang, "Collaborative distillation for ultra-resolution universal style transfer," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 1860–1869.

[4] T. Brooks, B. Mildenhall, T. Xue, J. Chen, D. Sharlet, and J. T. Barron, "Unprocessing images for learned raw denoising," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2019.

[5] K. Wei, Y. Fu, J. Yang, and H. Huang, "A physics-based noise formation model for extreme low-light raw denoising," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 2758–2767.

[6] L. Fan, F. Zhang, H. Fan, and C. Zhang, "Brief review of image denoising techniques," *Visual Computing for Industry, Biomedicine, and Art*, vol. 2, pp. 1–12, 2019.

[7] A. Kaur and G. Dong, "A complete review on image denoising techniques for medical images," *Neural Processing Letters*, vol. 55, no. 6, pp. 7807–7850, 2023.

[8] R. Kaur, G. Karmakar, and M. Imran, "Impact of traditional and embedded image denoising on cnn-based deep learning," *Applied Sciences*, vol. 13, no. 20, p. 11 560, 2023.

[9] S. Gupta *et al.*, "A review and comprehensive comparison of image denoising techniques," in *2014 International Conference on Computing for Sustainable Global Development*, 2014, pp. 972–976.

[10] Z. Kong, F. Deng, H. Zhuang, X. Yang, J. Yu, and L. He, "A comparison of image denoising methods," *arXiv preprint arXiv:2304.08990*, 2023.

[11] A. Lucas, M. Iliadis, R. Molina, and A. K. Katsaggelos, "Using deep neural networks for inverse problems in imaging: Beyond analytical methods," *IEEE Signal Processing Magazine*, vol. 35, no. 1, pp. 20–36, 2018.

[12] S. W. Hasinoff, D. Sharlet, R. Geiss, *et al.*, "Burst photography for high dynamic range and low-light imaging on mobile cameras," *ACM Transactions on Graphics*, vol. 35, no. 6, pp. 1–12, 2016.

[13] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[14] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge distillation: A survey," *International Journal of Computer Vision*, vol. 129, pp. 1789–1819, 2021.

[15] W. Chen, L. Peng, Y. Huang, M. Jing, and X. Zeng, "Knowledge distillation for u-net based image denoising," in *International Conference on ASIC*, 2021, pp. 1–4.

[16] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising," *IEEE transactions on image processing*, vol. 26, no. 7, pp. 3142–3155, 2017.

[17] V. Jain and S. Seung, "Natural image denoising with convolutional networks," *Advances in neural information processing systems*, vol. 21, 2008.

[18] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention*, 2015, pp. 234–241.

[19] Y.-L. Boureau, J. Ponce, and Y. LeCun, "A theoretical analysis of feature pooling in visual recognition," in *Proceedings of the 27th international conference on machine learning*, 2010, pp. 111–118.

[20] X. Mao, C. Shen, and Y.-B. Yang, "Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections," *Advances in neural information processing systems*, vol. 29, 2016.

[21] Y. Tai, J. Yang, X. Liu, and C. Xu, "Memnet: A persistent memory network for image restoration," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 4539–4547.

[22] C. Szegedy, W. Liu, Y. Jia, *et al.*, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[23] H. Chen, Y. Wang, T. Guo, *et al.*, "Pre-trained image processing transformer," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 12 299–12 310.

[24] Y. Wang, H. Huang, Q. Xu, J. Liu, Y. Liu, and J. Wang, "Practical deep raw image denoising on mobile devices," in *European Conference on Computer Vision*, Springer, 2020, pp. 1–16.

[25] J. Liu, C.-H. Wu, Y. Wang, *et al.*, *Learning raw image denoising with bayer pattern unification and bayer preserving augmentation*, 2019. arXiv: 1904.12945 [cs.CV].

[26] A. Beck and M. Teboulle, "Fast gradient-based algorithms for constrained total variation image denoising and deblurring problems," *IEEE transactions on image processing*, vol. 18, no. 11, pp. 2419–2434, 2009.

[27] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Physica D: Nonlinear phenomena*, vol. 60, no. 1-4, pp. 259–268, 1992.

[28] R. H. Chan, T. F. Chan, and C.-K. Wong, "Cosine transform based preconditioners for total variation deblurring," *IEEE transactions on Image Processing*, vol. 8, no. 10, pp. 1472–1478, 1999.

[29] C. R. Vogel and M. E. Oman, "Fast total variation-based image reconstruction," in *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, vol. 97669, 1995, pp. 1009–1015.

[30] N. Wiener, *Extrapolation, interpolation, and smoothing of stationary time series: with engineering applications*. The MIT press, 1949.

[31] P. Bouboulis, K. Slavakis, and S. Theodoridis, "Adaptive kernel-based image denoising employing semi-parametric regularization," *IEEE Transactions on Image Processing*, vol. 19, no. 6, pp. 1465–1479, 2010.

[32] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *International conference on computer vision*, 1998, pp. 839–846.

[33] G.-Z. Yang, P. Burger, D. N. Firmin, and S. Underwood, "Structure adaptive anisotropic image filtering," *Image and Vision Computing*, vol. 14, no. 2, pp. 135–145, 1996.

[34] H. Takeda, S. Farsiu, and P. Milanfar, "Kernel regression for image processing and reconstruction," *IEEE Transactions on image processing*, vol. 16, no. 2, pp. 349–366, 2007.

[35] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3-d transform-domain collaborative filtering," *IEEE Transactions on Image Processing*, vol. 16, no. 8, pp. 2080–2095, 2007. DOI: 10.1109/TIP.2007.901238.

[36] J. Portilla, V. Strela, M. J. Wainwright, and E. P. Simoncelli, "Image denoising using scale mixtures of gaussians in the wavelet domain," *IEEE Transactions on Image processing*, vol. 12, no. 11, pp. 1338–1351, 2003.

[37] M. Lebrun, "An Analysis and Implementation of the BM3D Image Denoising Method," *Image Processing On Line*, vol. 2, pp. 175–213, 2012.

[38] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Bm3d image denoising with shape-adaptive principal component analysis," in *Signal Processing with Adaptive Sparse Structured Representations*, 2009.

[39] M. Maggioni, V. Katkovnik, K. Egiazarian, and A. Foi, "Nonlocal transform-domain filter for volumetric data denoising and reconstruction," *IEEE transactions on image processing*, vol. 22, no. 1, pp. 119–133, 2012.

[40] M. Makitalo and A. Foi, "Optimal inversion of the anscombe transformation in low-count poisson image denoising," *IEEE transactions on Image Processing*, vol. 20, no. 1, pp. 99–109, 2010.

[41] L. Zhang, W. Dong, D. Zhang, and G. Shi, "Two-stage image denoising by principal component analysis with local pixel grouping," *Pattern recognition*, vol. 43, no. 4, pp. 1531–1549, 2010.

[42] M. Elad, B. Kawar, and G. Vaksman, "Image denoising: The deep learning revolution and beyond—a survey paper," *SIAM Journal on Imaging Sciences*, vol. 16, no. 3, pp. 1594–1654, 2023.

[43] Z. Wang, D. Liu, J. Yang, W. Han, and T. Huang, "Deep networks for image super-resolution with sparse prior," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 370–378.

[44] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*, 2015, pp. 448–456.

[45] C. Tang, L. Yuan, and P. Tan, "Lsm: Learning subspace minimization for low-level vision," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 6235–6246.

[46] P. Werbos, "Beyond regression: New tools for prediction and analysis in the behavioral sciences," 1974.

[47] P. J. Werbos, "Backpropagation through time: What it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[48] M. P. Reymann, T. Würfl, P. Ritt, *et al.*, "U-net for spect image denoising," in *IEEE Nuclear Science Symposium and Medical Imaging Conference*, 2019, pp. 1–2.

[49] L. Bao, Z. Yang, S. Wang, D. Bai, and J. Lee, "Real image denoising based on multi-scale residual dense block and cascaded u-net with block-connection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 448–449.

[50] S. Lee, M. Negishi, H. Urakubo, H. Kasai, and S. Ishii, "Mu-net: Multi-scale u-net for two-photon microscopy image denoising and restoration," *Neural Networks*, vol. 125, pp. 92–103, 2020.

[51] J. Gurrola-Ramos, O. Dalmau, and T. E. Alarcón, "A residual dense u-net neural network for image denoising," *IEEE Access*, vol. 9, pp. 31 742–31 754, 2021.

[52] M. Drozdzal, E. Vorontsov, G. Chartrand, S. Kadoury, and C. Pal, "The importance of skip connections in biomedical image segmentation," in *International Workshop on Deep Learning in Medical Image Analysis, International Workshop on Large-Scale Annotation of Biomedical Data and Expert Label Synthesis*, 2016, pp. 179–187.

[53] R. Komatsu and T. Gonsalves, "Comparing u-net based models for denoising color images," *AI*, vol. 1, no. 4, pp. 465–486, 2020.

[54] Z. Liu, Y. Lin, Y. Cao, *et al.*, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 10 012–10 022.

[55] C. Jia, Y. Yang, Y. Xia, *et al.*, "Scaling up visual and vision-language representation learning with noisy text supervision," in *International conference on machine learning*, 2021, pp. 4904–4916.

[56] T. Brown, B. Mann, N. Ryder, *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[57] X. Jia, J. Bartlett, T. Zhang, W. Lu, Z. Qiu, and J. Duan, "U-net vs transformer: Is u-net outdated in medical image registration?" In *International Workshop on Machine Learning in Medical Imaging*, 2022, pp. 151–160.

[58] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[59] J. Liang, J. Cao, G. Sun, K. Zhang, L. Van Gool, and R. Timofte, "Swinir: Image restoration using swin transformer," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 1833–1844.

[60] T. Plotz and S. Roth, "Benchmarking denoising algorithms with real photographs," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1586–1595.

[61] C. Chen, Q. Chen, J. Xu, and V. Koltun, "Learning to see in the dark," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 3291–3300.

[62] Q. Chen, J. Xu, and V. Koltun, "Fast image processing with fully-convolutional networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2497–2506.

[63] Y. Chen and T. Pock, "Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 6, pp. 1256–1272, 2016.

[64] S. Anwar and N. Barnes, "Real image denoising with feature attention," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 3155–3164.

[65] S. W. Zamir, A. Arora, S. Khan, M. Hayat, F. S. Khan, and M.-H. Yang, "Restormer: Efficient transformer for high-resolution image restoration," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 5728–5739.

[66] A. Abdelhamed, S. Lin, and M. S. Brown, "A high-quality denoising dataset for smartphone cameras," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 1692–1700.

[67] S. McKnight, S. G. Pierce, E. Mohseni, *et al.*, "Gans and alternative methods of synthetic noise generation for domain adaption of

defect classification of non-destructive ultrasonic testing," *arXiv preprint arXiv:2306.01469*, 2023.

[68] K. Zhang, Y. Li, J. Liang, *et al.*, "Practical blind image denoising via swin-conv-unet and data synthesis," *Machine Intelligence Research*, vol. 20, no. 6, pp. 822–836, 2023.

[69] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, Springer, 2016, pp. 630–645.

[70] C. Yang, X. Yu, Z. An, and Y. Xu, "Categories of response-based, feature-based, and relation-based knowledge distillation," in *Advancements in Knowledge Distillation: Towards New Horizons of Intelligent Systems*, Springer, 2023, pp. 1–32.

[71] J. Lehtinen, J. Munkberg, J. Hasselgren, *et al.*, "Noise2noise: Learning image restoration without clean data," *arXiv preprint arXiv:1803.04189*, 2018.

[72] L. D. Young, F. A. Reda, R. Ranjan, *et al.*, "Feature-align network with knowledge distillation for efficient denoising," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2022, pp. 709–718.

[73] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[74] Z. Li, Y. Wang, and J. Zhang, "Low-light image enhancement with knowledge distillation," *Neurocomputing*, vol. 518, pp. 332–343, 2023.

[75] X. Qin, Z. Wang, Y. Bai, X. Xie, and H. Jia, "Ffa-net: Feature fusion attention network for single image dehazing," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, 2020, pp. 11 908–11 915.

[76] W. Chen, Y. Huang, M. Wang, X. Wu, and X. Zeng, "Tsdn: Two-stage raw denoising in the dark," *IEEE Transactions on Image Processing*, 2023.

[77] Y. Chi, A. Gnanasambandam, V. Koltun, and S. H. Chan, "Dynamic low-light imaging with quanta image sensors," in *European Conference, Glasgow, UK*, 2020, pp. 122–138.

[78] Y. Jiang, J. Nawala, F. Zhang, and D. Bull, "Compressing deep image super-resolution models," *arXiv preprint arXiv:2401.00523*, 2023.

[79] J. Xie, L. Gong, S. Shao, S. Lin, and L. Luo, "Hybrid knowledge distillation from intermediate layers for efficient single image super-resolution," *Neurocomputing*, vol. 554, p. 126 592, 2023.

[80] T. Choudhary, V. Mishra, A. Goswami, and J. Sarangapani, "A comprehensive survey on model compression and acceleration," *Artificial Intelligence Review*, vol. 53, pp. 5113–5155, 2020.

[81] Z. Chen, Y. Zhang, J. Gu, y. zhang yongbing, L. Kong, and X. Yuan, "Cross aggregation transformer for image restoration," in *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., 2022, pp. 25 478–25 490.

[82] M. Anderson, R. Motta, S. Chandrasekar, and M. Stokes, "Proposal for a standard default color space for the internet—srgb," in *Color and imaging conference*, Society of Imaging Science and Technology, vol. 4, 1996, pp. 238–245.

[83] K. Han, A. Xiao, E. Wu, J. Guo, C. Xu, and Y. Wang, "Transformer in transformer," *Advances in neural information processing systems*, vol. 34, pp. 15 908–15 919, 2021.

[84] K. Ma and Y. Fang, "Image quality assessment in the modern age," in *Proceedings of the 29th ACM International Conference on Multimedia*, 2021, pp. 5664–5666.

[85] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne.," *Journal of machine learning research*, vol. 9, no. 11, 2008.

[86] L. Van Der Maaten, "Accelerating t-sne using tree-based algorithms," *The journal of machine learning research*, vol. 15, no. 1, pp. 3221–3245, 2014.

[87] M. Wattenberg, F. Viégas, and I. Johnson, "How to use t-sne effectively," *Distill*, vol. 1, no. 10, e2, 2016.

[88] Y. Li, Y. Zhang, R. Timofte, *et al.*, "Ntire challenge on image denoising: Methods and results," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 1904–1920.

[89] X. Chen, Q. Cao, Y. Zhong, J. Zhang, S. Gao, and D. Tao, "Dearkd: Data-efficient early knowledge distillation for vision transformers," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 12 052–12 062.

[90] S. S. Stevens, "On the psychophysical law.," *Psychological review*, vol. 64, no. 3, p. 153, 1957.

[91] D.-T. Dang-Nguyen, C. Pasquini, V. Conotter, and G. Boato, "Raise: A raw images dataset for digital image forensics," in *Proceedings of the 6th ACM multimedia systems conference*, 2015, pp. 219–224.

[92] V. Bychkovsky, S. Paris, E. Chan, and F. Durand, "Learning photographic global tonal adjustment with a database of input / output image pairs," in *The Twenty-Fourth IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[93] K. Cabello-Solorzano, I. Ortigosa de Araujo, M. Peña, L. Correia, and A. J. Tallón-Ballesteros, "The impact of data normalization on the accuracy of machine learning algorithms: A comparative analysis," in *International Conference on Soft Computing Models in Industrial and Environmental Applications*, Springer, 2023, pp. 344–353.

[94] N. Anantrasirichai and D. Bull, "Artificial intelligence in the creative industries: A review," *Artificial intelligence review*, vol. 55, no. 1, pp. 589–656, 2022.

[95] B. Mildenhall, P. Hedman, R. Martin-Brualla, P. P. Srinivasan, and J. T. Barron, "Nerf in the dark: High dynamic range view synthesis from noisy raw images," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 16 190–16 199.

[96] H. Qassim, A. Verma, and D. Feinzimer, "Compressed residual-vgg16 cnn model for big data places image recognition," 2018, pp. 169–175. DOI: 10.1109/CCWC.2018.8301729.

[97] D. Theckedath and R. Sedamkar, "Detecting affect states using vgg16, resnet50 and se-resnet50 networks," *SN Computer Science*, vol. 1, no. 2, p. 79, 2020.

[98] S. Guo, Z. Liang, and L. Zhang, "Joint denoising and demosaicking with green channel prior for real-world burst images," *IEEE Transactions on Image Processing*, vol. 30, pp. 6930–6942, 2021.

[99] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus, "Deconvolutional networks," in *2010 IEEE Computer Society Conference on computer vision and pattern recognition*, IEEE, 2010, pp. 2528–2535.

[100] X. Wang, L. Xie, C. Dong, and Y. Shan, "Real-esrgan: Training real-world blind super-resolution with pure synthetic data," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 1905–1914.

[101] A. Khattar and S. Quadri, ""generalization of convolutional network to domain adaptation network for classification of disaster images on twitter"," *Multimedia Tools and Applications*, vol. 81, Sep. 2022. DOI: 10.1007/s11042-022-12869-1.

[102] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[103] A. Bagaskara and M. Suryanegara, "Evaluation of vgg-16 and vgg-19 deep learning architecture for classifying dementia people," in *4th International Conference of Computer and Informatics Engineering (IC2IE)*, IEEE, 2021, pp. 1–4.

[104] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *Computer Vision–ECCV: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14*, Springer, 2016, pp. 694–711.

[105] T. Wu, Q. Huang, Z. Liu, Y. Wang, and D. Lin, "Distribution-balanced loss for multi-label classification in long-tailed datasets," in *16th European Conference, Proceedings, Part IV 16*, Springer, 2020, pp. 162–178.

[106] A. Horé and D. Ziou, "Image quality metrics: Psnr vs. ssim," 2010, pp. 2366–2369. DOI: 10.1109/ICPR.2010.579.

[107] S. Mehta, C. Paunwala, and B. Vaidya, "Cnn based traffic sign classification using adam optimizer," in *international conference on in-*

telligent computing and control systems (ICCS), IEEE, 2019, pp. 1293–1298.

[108] K. Janocha and W. M. Czarnecki, "On loss functions for deep neural networks in classification," *arXiv preprint arXiv:1702.05659*, 2017.

[109] X. He and J. Cheng, "Revisiting l1 loss in super-resolution: A probabilistic view and beyond," *arXiv preprint arXiv:2201.10084*, 2022.

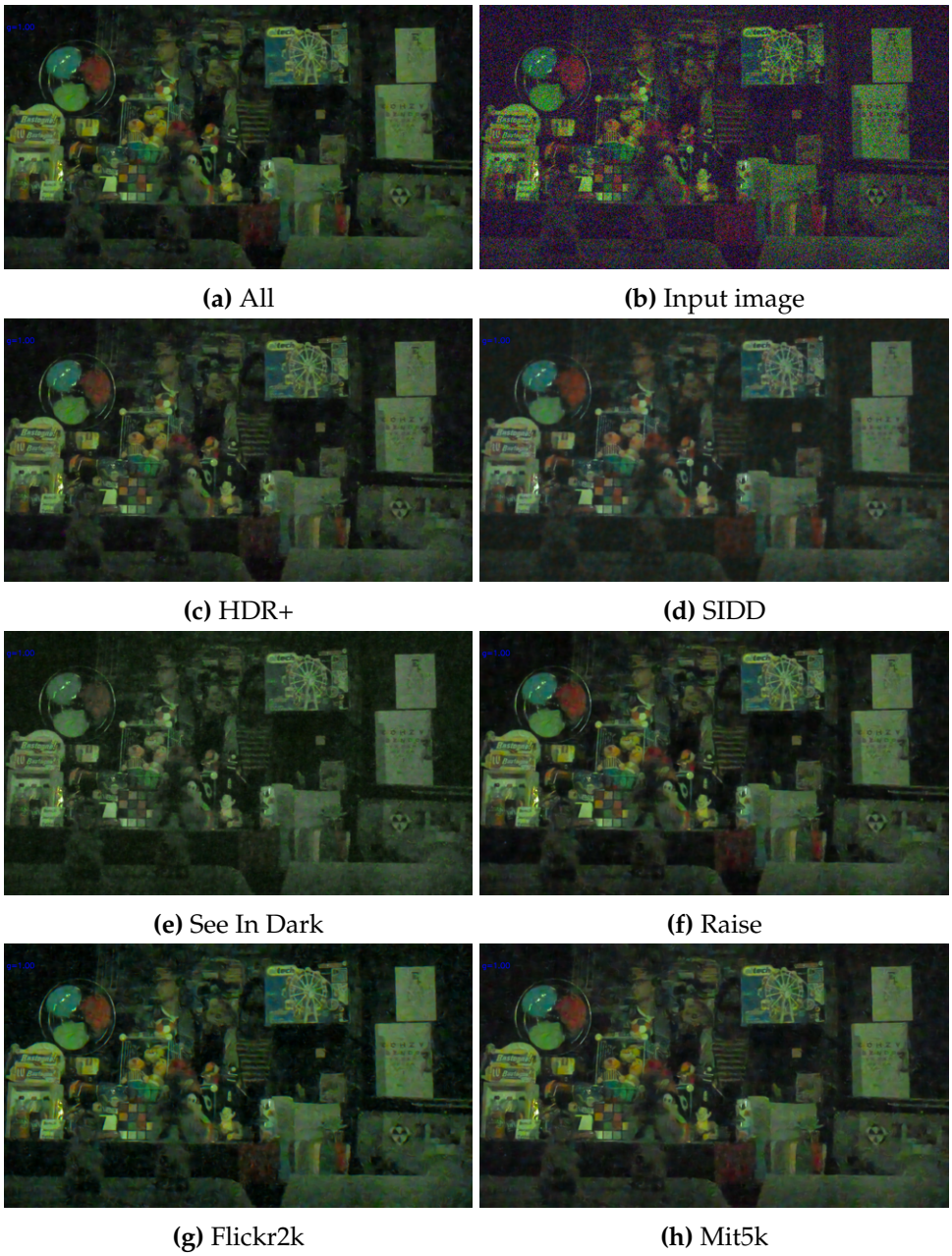
[110] D. R. I. M. Setiadi, "Psnr vs ssim: Imperceptibility quality assessment for image steganography," *Multimedia Tools and Applications*, vol. 80, no. 6, pp. 8423–8444, 2021.

[111] A. Acharya and J. Ortner, "Progressive learning," *Econometrica*, vol. 85, no. 6, pp. 1965–1990, 2017.

[112] G. W. Anderson and D. J. Castano, "Measures of fine tuning," *Physics Letters B*, vol. 347, no. 3-4, pp. 300–308, 1995.

A. Data Selection Results

A.1 Separate Datasets Single Frame



(a) All **(b)** Input image

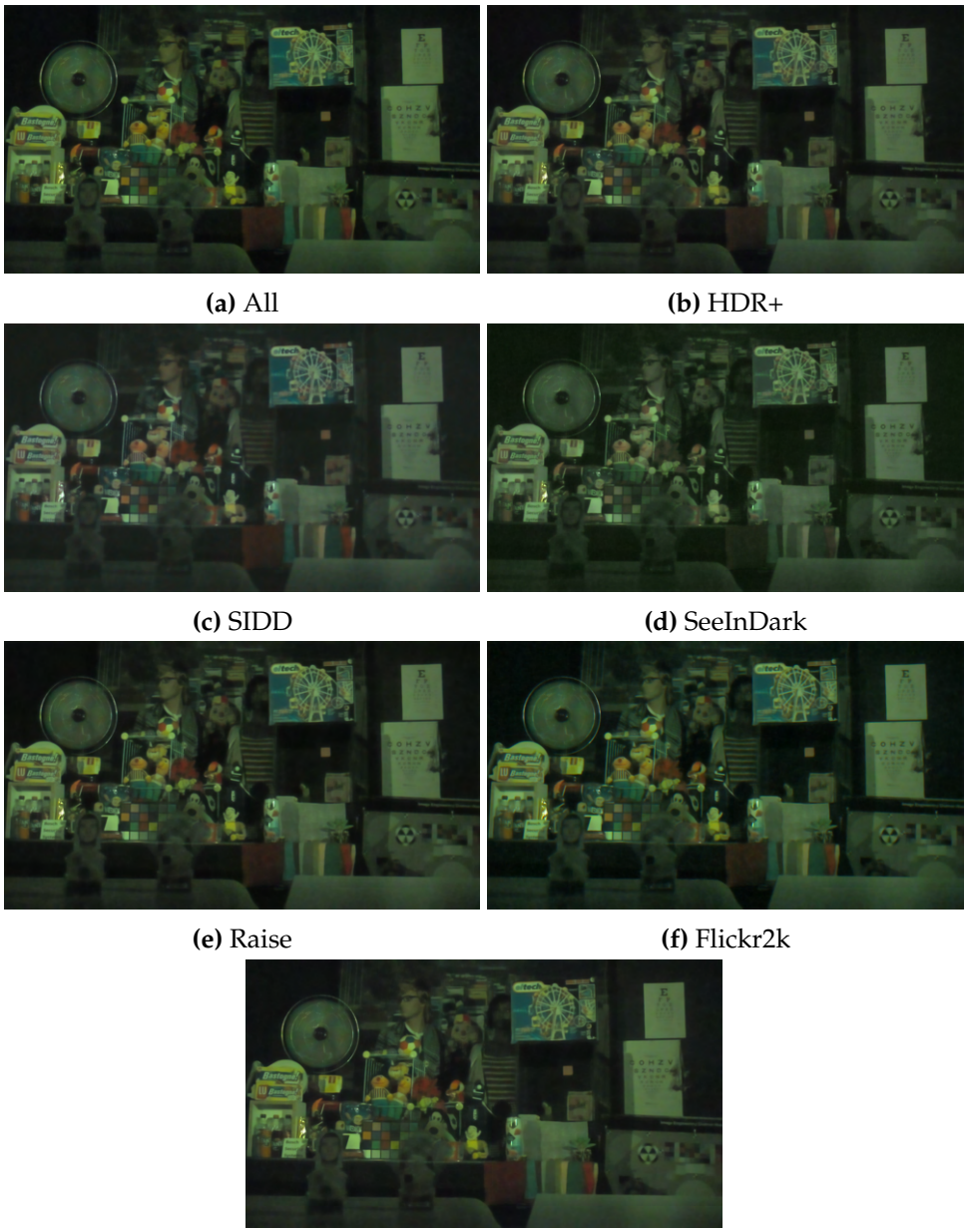
(c) HDR+ **(d)** SIDD

(e) See In Dark **(f)** Raise

(g) Flickr2k **(h)** Mit5k

Figure A.1: Separate U-net trained models tested on target test set with single frames containing 60 db noise

A.2 Separate Datasets Averaged



(a) All **(b)** HDR+

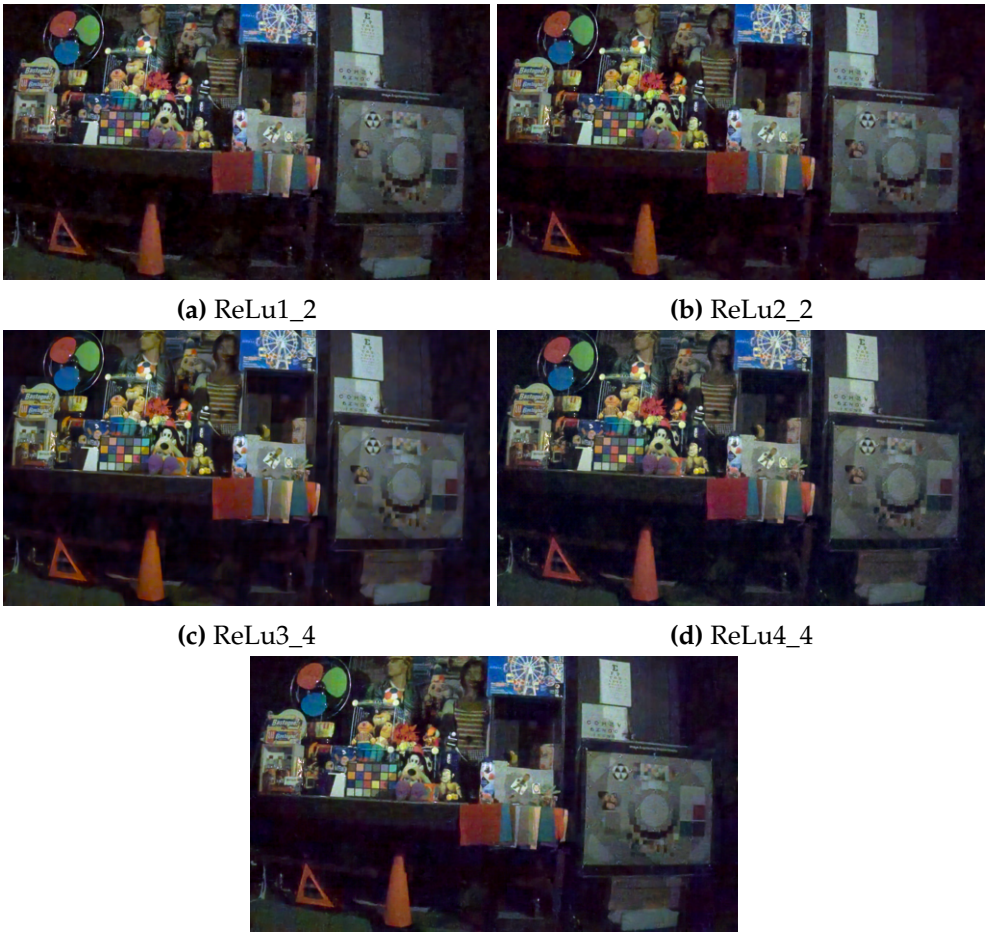
(c) SIDD **(d)** SeeInDark

(e) Raise **(f)** Flickr2k

(g) Mit5k

Figure A.2: Separate U-net trained models tested on target test set averaged over 100 frames with 60 db noise.

B. Grid Artifact Abblation



(a) ReLu1_2 **(b)** ReLu2_2

(c) ReLu3_4 **(d)** ReLu4_4

(e) ReLu5_4

Figure B.1: SCUNet trained with separate layers for perceptual loss, based on 63db noise

C. Teacher Ablation Qualitative Results for Restormer-based Models

Figure C.1: R01: Baseline Restormer model with Conv transpose upsampling, progressive learning. Result on 32dB noise real sensor data.

Figure C.2: R02: Restormer model with the loss based on SCUNet loss, with Conv transpose upsampling, progressive learning. Result on 32dB noise real sensor data.

Figure C.3: R03: Restormer model with the overall skip connection, the loss based on SCUNet loss, Conv transpose upsampling, progressive learning. Result on 32dB noise real sensor data.

Figure C.4: R04: Restormer model without progressive learning, with the loss based on SCUNet loss, with Conv transpose upsampling. Result on 32dB noise real sensor data.

Figure C.5: R05: Restormer model with the noise trained until 1024, the loss based on SCUNet loss, Conv transpose upsampling, progressive learning. Result on 32dB noise real sensor data.

Figure C.6: R06: Restormer model with wgan as adversarial loss, the loss based on SCUNet loss, Conv transpose upsampling, progressive learning. Result on 32dB noise real sensor data.

Figure C.7: R01: Baseline Restormer model with Conv transpose upsampling, progressive learning. Result on 45dB noise real sensor data.

Figure C.8: R02: Restormer model with the loss based on SCUNet loss, with Conv transpose upsampling, progressive learning. Result on 45dB noise real sensor data.

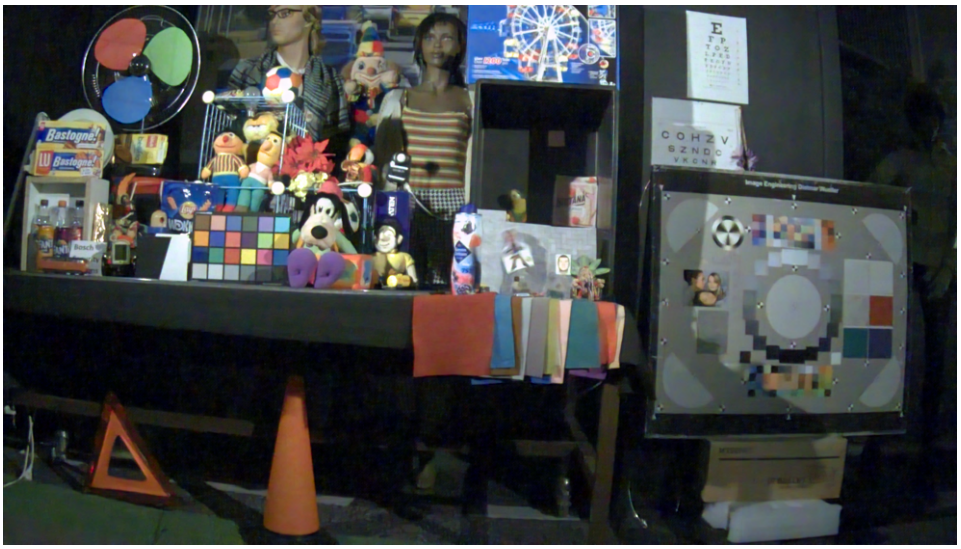


Figure C.9: R03: Restormer model with the overall skip connection, the loss based on SCUNet loss, Conv transpose upsampling, progressive learning. Result on 45dB noise real sensor data.

Figure C.10: R04: Restormer model without progressive learning, with the loss based on SCUNet loss, with Conv transpose upsampling. Result on 45dB noise real sensor data.

Figure C.11: R05: Restormer model with the noise trained until 1024, the loss based on SCUNet loss, Conv transpose upsampling, progressive learning. Result on 45dB noise real sensor data.

Figure C.12: R06: Restormer model with wgan as adversarial loss, the loss based on SCUNet loss, Conv transpose upsampling, progressive learning. Result on 45dB noise real sensor data.

Figure C.13: R01: Baseline Restormer model with Conv transpose upsampling, progressive learning. Result on 63dB noise real sensor data.

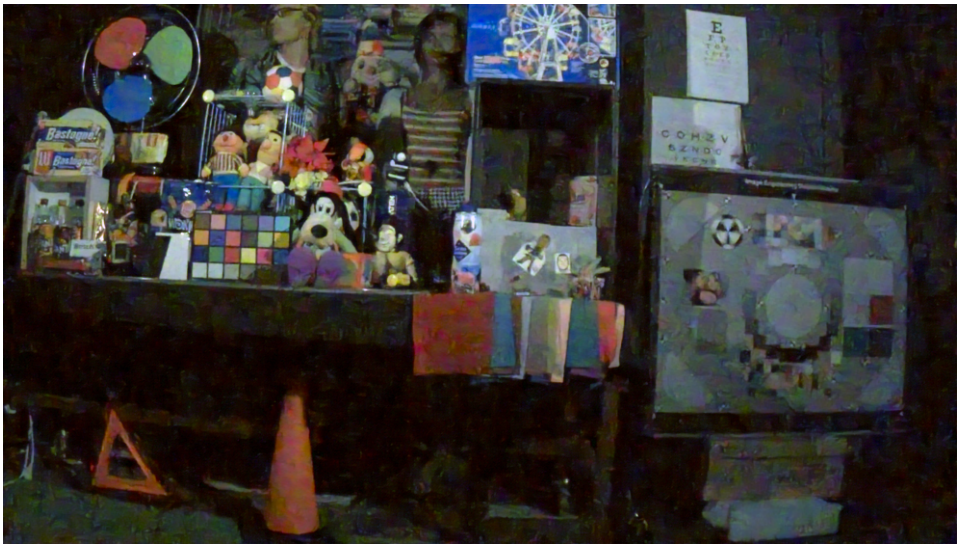


Figure C.14: R02: Restormer model with the loss based on SCUNet loss, with Conv transpose upsampling, progressive learning. Result on 63dB noise real sensor data.

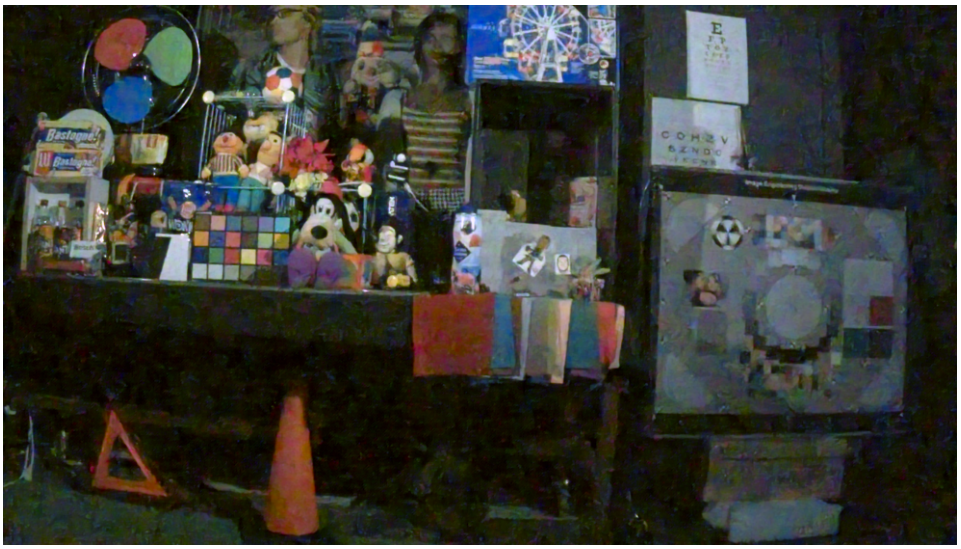


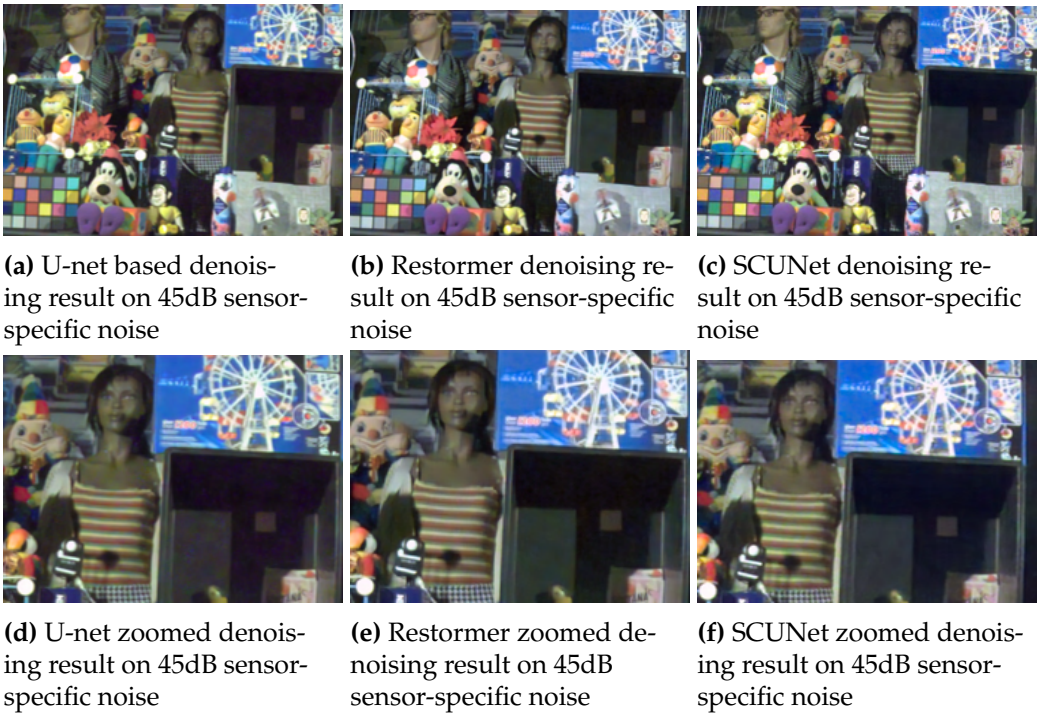
Figure C.15: R03: Restormer model with the overall skip connection, the loss based on SCUNet loss, Conv transpose upsampling, progressive learning. Result on 63dB noise real sensor data.

Figure C.16: R04: Restormer model without progressive learning, with the loss based on SCUNet loss, with Conv transpose upsampling. Result on 63dB noise real sensor data.

Figure C.17: R05: Restormer model with the noise trained until 1024, the loss based on SCUNet loss, Conv transpose upsampling, progressive learning. Result on 63dB noise real sensor data.

Figure C.18: R06: Restormer model with wgan as adversarial loss, the loss based on SCUNet loss, Conv transpose upsampling, progressive learning. Result on 63dB noise real sensor data.

D. Teacher Comparison Qualitative Results



(a) U-net based denoising result on 45dB sensor-specific noise

(b) Restormer denoising result on 45dB sensor-specific noise

(c) SCUNet denoising result on 45dB sensor-specific noise

(d) U-net zoomed denoising result on 45dB sensor-specific noise

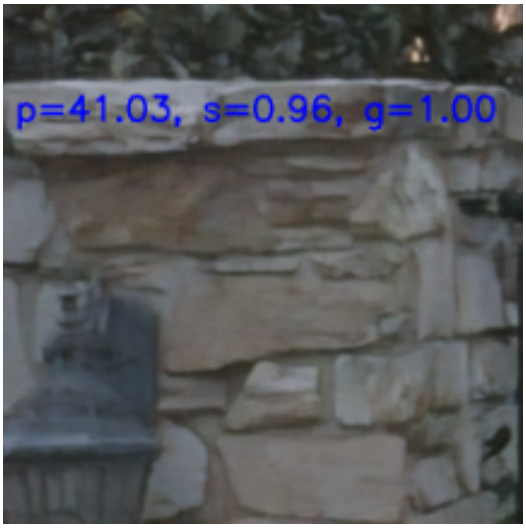
(e) Restormer zoomed denoising result on 45dB sensor-specific noise

(f) SCUNet zoomed denoising result on 45dB sensor-specific noise

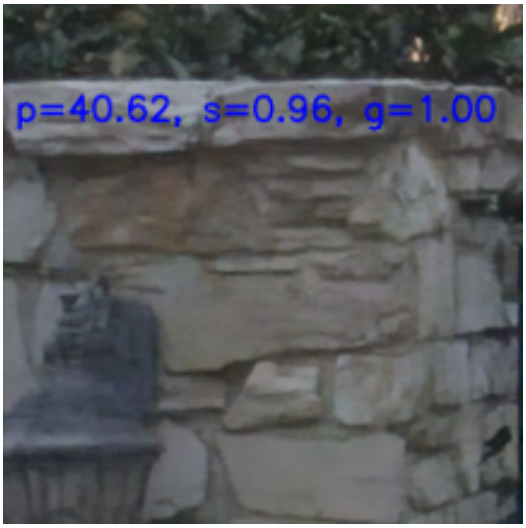
Figure D.1: Teacher comparison results on 45dB sensor-specific data

(a) Unified testset input image containing 45dB noise

(b) U-net denoising result on 45dB of noise added to unified testset



(c) Restormer denoising result on 45dB of noise added to unified testset



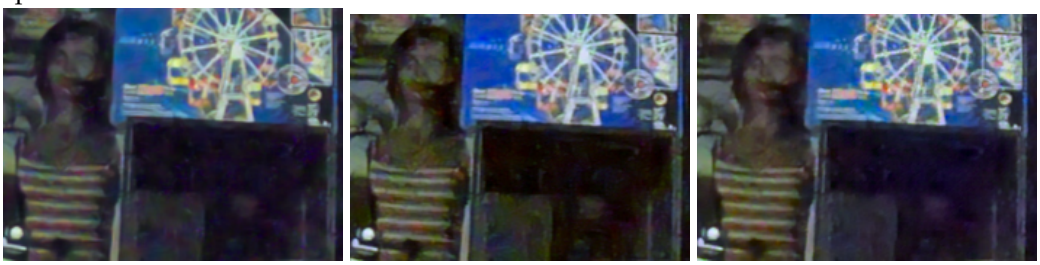
(d) SCUNet denoising result on 45dB of noise added to unified testset

Figure D.2: Teacher comparison results on unified testset with 45dB noise

(a) U-net based denoising result on 63dB sensor-specific noise

(b) Restormer denoising result on 63dB sensor-specific noise

(c) SCUNet denoising result on 63dB sensor-specific noise

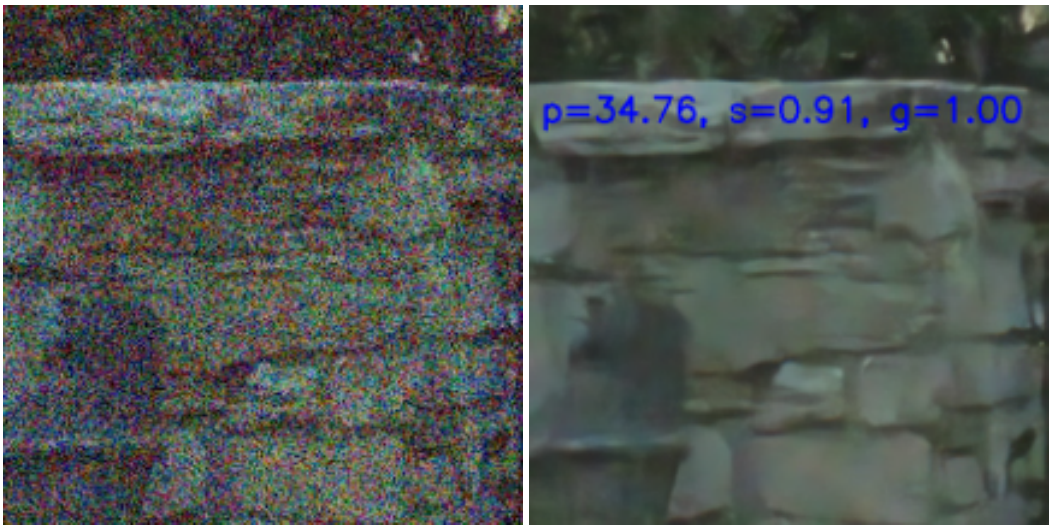


(d) U-net zoomed denoising result on 63dB sensor-specific noise

(e) Restormer zoomed denoising result on 63dB sensor-specific noise

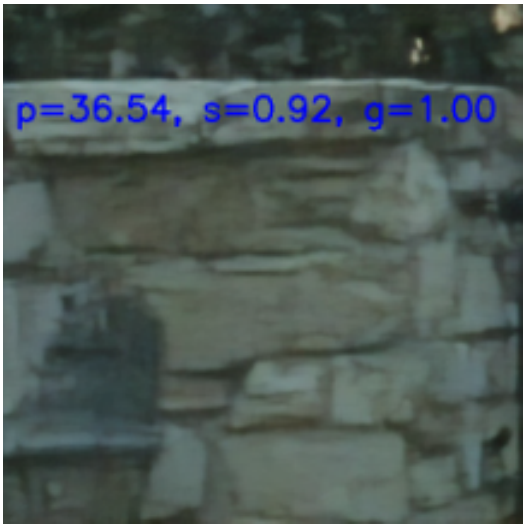
(f) SCUNet zoomed denoising result on 63dB sensor-specific noise

Figure D.3: Teacher comparison results on 63dB sensor-specific data



(a) Unified testset input image containing 63dB noise

(b) U-net denoising result on 63dB of noise added to unified testset



(c) Restormer denoising result on 63dB of noise added to unified testset

(d) SCUNet denoising result on 63dB of noise added to unified testset

Figure D.4: Teacher comparison results on unified testset with 63dB noise

E. Student Ablation Qualitative Results using Knowledge Distillation

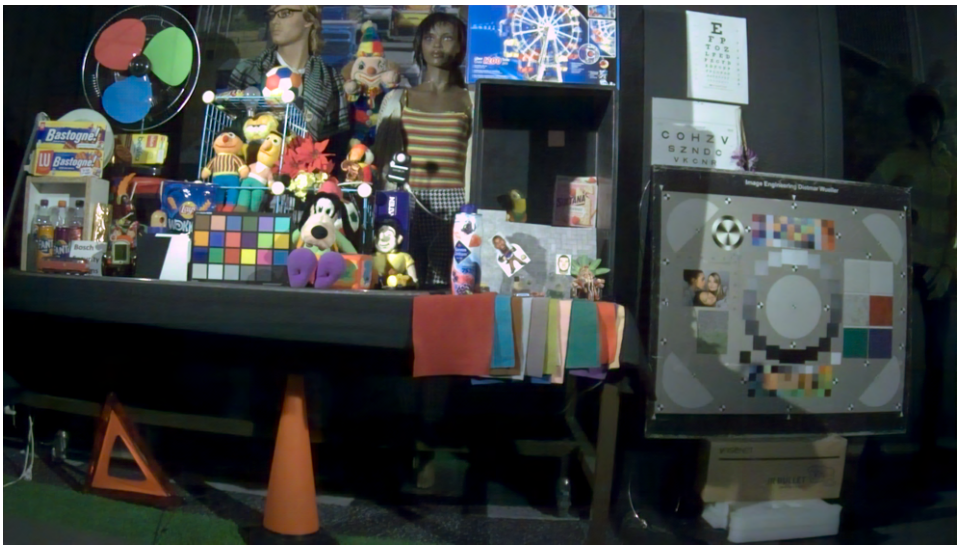


Figure E.1: KD01: Compressed SCUNet model using 2 blocks for each resolution step, trained from scratch. Result on 32dB noise real sensor data.

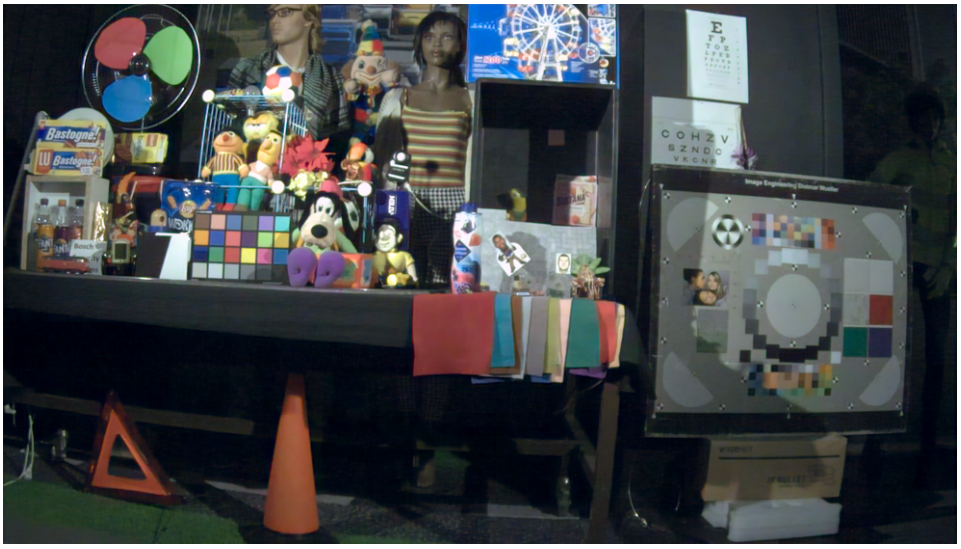


Figure E.2: KD02: Compressed SCUNet model using 1 blocks for each resolution step, trained from scratch. Result on 32dB noise real sensor data.

Figure E.3: KD03: Compressed SCUNet model using 2 blocks for each resolution step, with response-based knowledge distillation (w=1) from S05. Result on 32dB noise real sensor data.

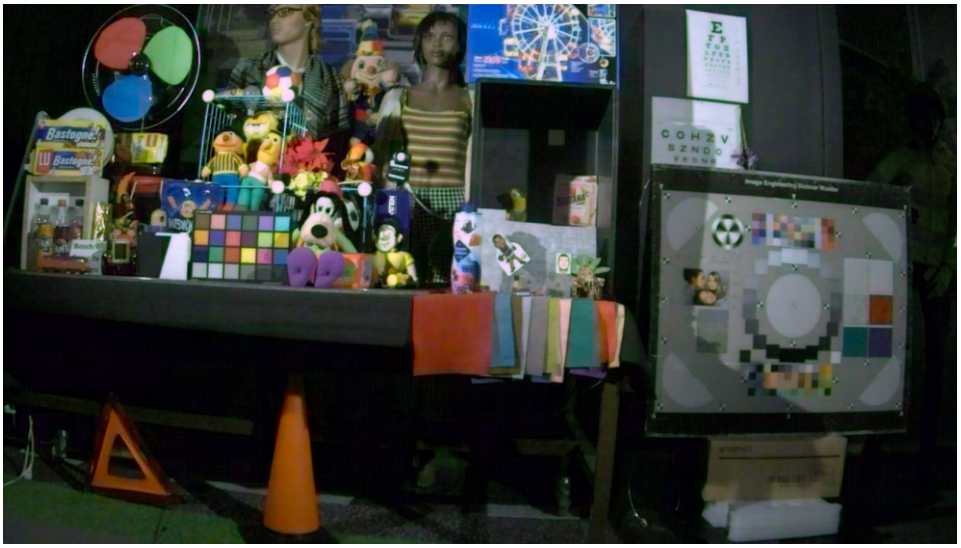


Figure E.4: KD04: Compressed SCUNet model using 1 blocks for each resolution step, with response-based knowledge distillation (w=1) from S05. Result on 32dB noise real sensor data.

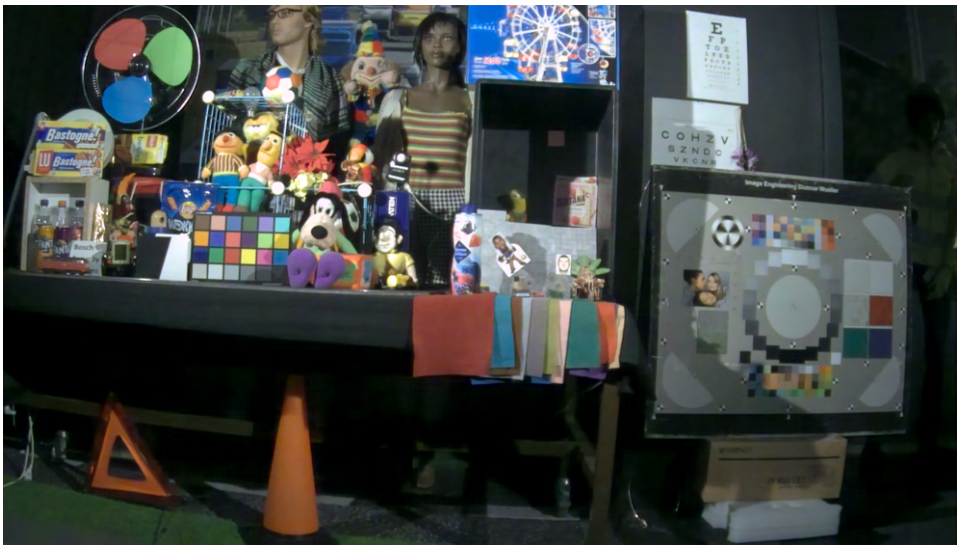


Figure E.5: KD05: Compressed SCUNet model using 2 blocks for each resolution step, with feature-based knowledge distillation (w=1) from S05. Result on 32dB noise real sensor data.

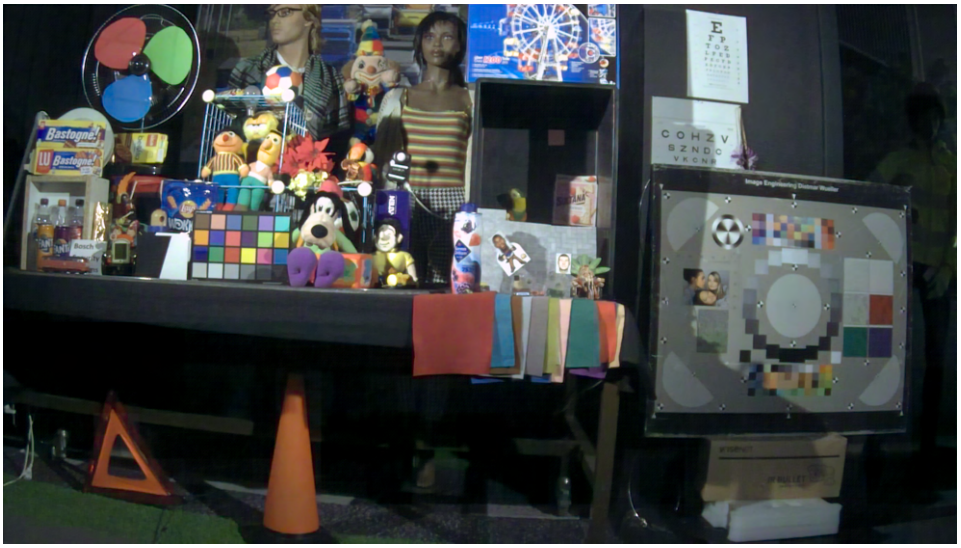


Figure E.6: KD06: Compressed SCUNet model using 2 blocks for each resolution step, with feature-based knowledge distillation (w=1) from S05. Result on 32dB noise real sensor data.

Figure E.7: KD07: Compressed SCUNet model using 2 blocks for each resolution step, with feature-based and response-based knowledge distillation (w=10) from S05. Result on 32dB noise real sensor data.

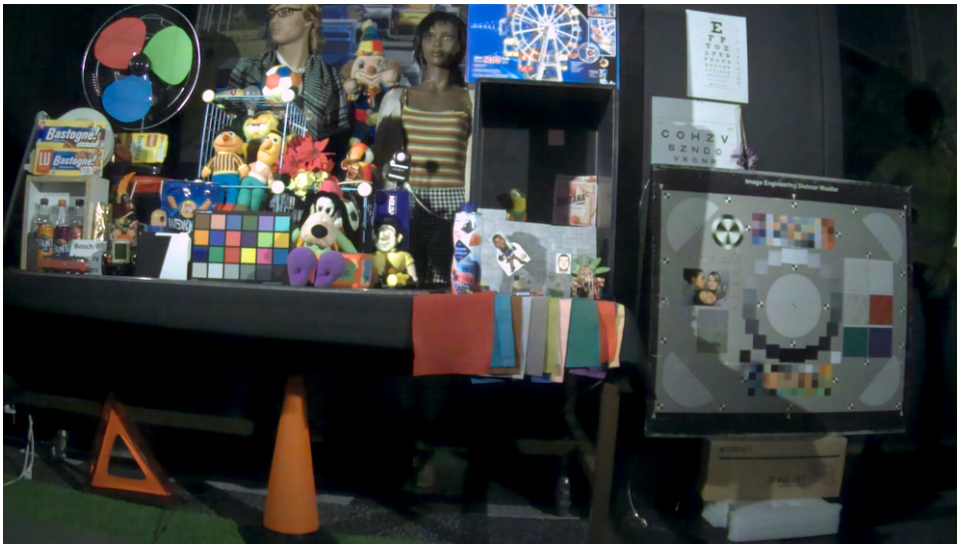


Figure E.8: KD08: Compressed SCUNet model using 2 blocks for each resolution step, with feature-based and response-based knowledge distillation (w=0.1) from S05. Result on 32dB noise real sensor data.