BRNO UNIVERSITY OF TECHNOLOGY

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF TELECOMMUNICATIONS

ÚSTAV TELEKOMUNIKACÍ

RAW IMAGE DEBAYERIZATION USING DEEP NEURAL NETWORK

DEBAYERIZACE RAW FOTOGRAFIE POMOCÍ HLUBOKÉ NEURONOVÉ SÍTĚ

BACHELOR'S THESIS

BAKALÁŘSKÁ PRÁCE

AUTHOR
AUTOR PRÁCE

Peter Balušík

SUPERVISOR
VEDOUCÍ PRÁCE

prof. Mgr. Pavel Rajmic, Ph.D.

BRNO 2023

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

Bachelor's Thesis

Bachelor's study program **Telecommunication and Information Systems**

Department of Telecommunications

Student:	Peter Balušík	***ID:***	230531
Year of study:	3	***Academic year:***	2022/23

TITLE OF THESIS:

RAW image debayerization using deep neural network

INSTRUCTION:

The student will learn about the physics of capturing light on the digital camera sensor and about the Bayer mask, which enables the acquisition of an image in its RGB components. He will also study the principle of debayerization/demosaicing [1], where individual spatially separated RGB values are interpolated into the format of an usual square grid of pixels. Therefore, affter demosaicing, each pixel already "contains" the RGB values at the same spatial coordinate. The student will also become familiar with the methods that are commonly used for demosaicing. The work will then focus on studying and understanding the principle of the neural encoder based on the "deep image prior" concept [2]. The neural network will be adapted for the problem of demosaicing a RAW photo. The student will test the network, and he will qualitatively and quantitatively evaluate the outputs, also with respect to usual demosaicing methods.

RECOMMENDED LITERATURE:

[1] MENON, Daniele a Giancarlo CALVAGNO. Color image demosaicking: An overview. Signal processing. Image communication. Amsterdam: Elsevier B.V, 2011, 26(8), 518-533. ISSN 0923-5965. doi:10.1016/j.image.2011.04.003

[2] ULYANOV, Dmitry, Andrea VEDALDI a Victor LEMPITSKY. Deep Image Prior. International journal of computer vision. New York: Springer US, 2020, 128(7), 1867-1888. ISSN 0920-5691. doi:10.1007/s11263-020-01303-4

Date of project specification:	6.2.2023	***Deadline for submission:***	26.5.2023

Supervisor: prof. Mgr. Pavel Rajmic, Ph.D.

prof. Ing. Jiří Mišurec, CSc.
Chair of study program board

ABSTRACT

This thesis focuses on the problem of demosaicing; specifically, demosaicing using deep image prior. Deep image prior (DIP) is a concept that uses untrained convolutional neural networks to solve common reconstruction problems, with the only input information being an image degraded in some way. The aim of this thesis is to find out whether the DIP is a viable method for demosaicing problems. A new demosaicing method based on DIP is proposed and compared with common demosaicing methods. Different color filer arrays (CFAs) were tested to see the full potential of the proposed method. A numerical comparison was made using a variety of assessment methods. Based on this comparison, the proposed method proved to be similar, in some cases even better than the widely used Malvar's demosaicing method. Visually, the proposed method displayed similar results to the finest method in the experiments – the Menon's demosaicing method. Additionally, averaging the last few images of the optimization process proved to bring positive results in terms of numerical comparison. Even though the proposed method brought some interesting results, it turned out to be extremely computationally challenging when compared with other common demosaicing methods.

KEYWORDS

artificial neural network, Bayer filter, color filter array, convolutional neural network, Deep Image Prior, demosaicing, image restoration, inpainting, Malvar's demosaicing method, Menon's demosaicing method, X-Trans filter

ABSTRAKT

Táto práca sa zaoberá problémom debayerizácie a to konkrétne debayerizáciou pomocou deep image prior. Deep image prior (DIP) je koncept riešenia bežných rekonštrukčných problémov použitím netrénovaných konvolučných neurónových sietí. Jedinou vstupnou informáciou je obrázok, ktorý bol nejakým spôsobom poškodený. Cieľom tejto práce je zistiť, či je DIP použiteľná metóda na problémy debayerizácie. Taktiež bola navrhnutá nová debayerizačná metóda založená na DIP a porovnaná s bežnými debayerizačnými metódami. Rôzne mozaikové farebné filtre (CFAs) boli otestované na zistenie plného potenciálu navrhnutej metódy. Číselné porovnanie bolo spravené použitím rôznych metód hodnotenia. Na základne tohto porovnania, zvolená metóda preukázala podobné, v niektorých prípadoch aj lepšie, výsledky ako Malvarova debayerizačná metóda. Vizuálne, navrhovaná metóda ukázala podobné výsledky k najkvalitnejšej metóde v experimentoch – Menonovej debayerizačnej metóde. Dodatočne, spriemerovanie posledných pár obrázkov optimizačného procesu prinieslo pozitívne výsledky vzhľadom na číselné porovnanie. Aj keď navrhovaná metóda priniesla zaujímavé výsledky, ukázalo sa, že je mimoriadne výpočetne náročná v porovnaní s ďalšími bežnými debayerizačnými metódami.

KĽÚČOVÉ SLOVÁ

Bayerov filter, debayerizácia, Deep Image Prior, konvolučná neurónová sieť, Malvarova debayerizačná metóda, Menonova debayerizačná metóda, mozaikový farebný filter, rekonštrukcia obrazu, umelá neurónová sieť, vypĺňovanie obrazu, X-Trans filter

ROZŠÍŘENÝ ABSTRAKT

Prakticky každý človek spravil nejakú fotku vo svojom živote, či už to bolo s foťákom alebo mobilom. Len pár z nich by vedelo povedať ako sa táto fotka stala farebnou. Skrátene, digitálna fotka vznikne tým, že svetlo z vonkajšieho prostredia prejde objektívom kamery a zachytí sa na senzori v kamere. Tento senzor má zvyčajne na sebe nejaký farebný filter. Na to aby boli zachytené všetky farby (červená, zelená a modrá) by kamera musela obsahovať tri rôzne senzory s troma rôznymi farebnými filtrami. Toto riešenie by bolo veľmi zložité a finančne náročné, preto sa v bežných kamerách používa len jeden senzor s jedným špecialnym farebným filtrom. Tento filter sa nazýva mozaikový farebný filter (alebo CFA) a v spojení so senzorom v kamere tvoria CFA senzor.

Mozaikový farebný filter (CFA) je spravený s príncípom, že každá časť mozaiky prepustí inú zložku svetla, buď prepustí červenú, zelenú alebo modrú zložku. Týmto spôsobom sa zachytia jednotlivé zložky svetla na jednom CFA senzori. Na získanie všetkých zložiek na každej časti mozaiky sa používa debayerizácia (*demosaicing*). Debayerizácia je proces, pri ktorom sa odhadujú alebo dopočítavajú chýbajúce farebné zložky na mozaike. Tento proces sa uskutoční v kamere a spraví farebnú fotku, ktorá je viditeľná uživateľom. Ak by bola fotka vybratá pred týmto procesom, vznikla by fotka v RAW formáte. Niektorí ľudia (hlavne fotografovia) preferujú fotky v RAW formáte, pretože debayerizačný proces a úpravy môžu spraviť v nejakom lepšom softvéri napr. Adobe Lightroom.

Existuje veľa metód debayerizácie. Tá najjednoduchšia sa nazýva bilineárna interpolácia. Ďaľšie, viac efektívne, metódy používajú prepracovanejšie techniky, medzi tieto metódy patrí napr. Malvarova interpolačná metóda a Menonova debayerizačná metóda. Táto práca sa snaží implementovať iný prístup k debayerizácii a to pomocou hlbokej konvolučnej neurónovej siete. Cieľom tejto práce bolo implementovať debayerizačnú metódu na základne konceptu Deep Image Prior (DIP) a nasledovne túto metódu kvalitatívne a kvantitatívne porovnať s bežnými debayerizačnými metódami.

Táto práca sa skladá zo štyroch hlavných kapitol. V prvej kapitole sú opísané základy neurónových sietí a ich bežne používané architektúry. Druhá kapitola je zameraná na Deep Image Prior a stručne opisuje metódu používanú v DIP. Ďaľej sa zameriava na rôzne aplikácie navrhnuté v originálnom článku o DIP – odšumovanie, vyplňovanie obrazu, zväčšovanie rozlíšenia a ďaľšie. V tretej kapitole je bližšie opísaná debayerizácia, Bayerov filter a rôzne metódy používané na debayerizáciu. Taktiež popisuje čo sú a ako vznikajú RAW obrázky. Posledná (štvrtá) kapitola opisuje implementáciu navrhnutej debayerizačnej metódy, jej výhody a nevýhody, a následne porovnanie s bežne používanými metódami.

V rámci DIP sú debayerizácia (*demosaicing*) a vyplňovanie obrazu (*inpainting*)

veľmi podobné – obidve metódy sa snažia doplniť nejaké chýbajúce hodnoty pixelov. Avšak medzi nimi existuje aj niekoľko rozdielov. Hlavný rozdiel je v interpretácii chýbajúcich častí. Inpainting na to používa binárnu masku a debayerizácia využíva CFA masku. Na základe týchto vlastností bola navrhnutá aj metóda v tejto práci. Takisto s touto metódou bolo vyskúšaných aj viacero CFA masiek ako napr. Bayerova maska, náhodná maska a X-Trans filter.

Navrhovaná metóda bola porovnaná s týmito metódami – bilineárna interpolácia, Malvarova metóda, Menonova metóda a funkcia `demosaic` v Matlabe. Na základe číselného porovnania, pomocou rôznych spôsobov hodnotenia (PSNR, PSNR-HVS, PSNR-HVSM a SSIM), sa navrhovaná metóda podobala Malvarovej metóde. V prípade využitia CFA masky X-Trans dokonca prekonala Malvarovu metódu. Menonova metóda bola najkvalitnejšia vzhľadom na číselné porovnanie. Vizuálne sa navrhnutá metóda podobala Menonovej metóde, najlepšej metóde používanej v experimentoch. V rámci testovaných CFA masiek sa X-trans filter umiestil na prvom mieste, náhodná maska na druhom a Bayerov filter na poslednom.

Zaujímavé zistenie bolo to, že spriemerovanie posledných pár obrázkov optimizačného procesu neurónovej siete prinieslo pozitívne výsledky v rámci číselného porovnania (vizuálna kvalita sa nejako značne nezmenila). Dokonca toto spriemerovanie prinieslo lepšie výsledky ako takzvaný „oracle approach“.

Bežné metódy sú spravené len na jeden typ CFA masky a to väčšinou na Bayerovu masku. Okrem výborných vizuálnych výsledkov má navrhnutá metóda aj ďalšiu výhodu oproti bežným metódam – dá sa aplikovať na hocijakú CFA. Napriek týmto výhodám má spomínaná metóda jednu veľkú nevýhodu a tou je jej výpočetná náročnosť. Vďaka tejto veľkej výpočetnej náročnosti by navrhnutá metóda bola tažšie využiteľná v praxi.

BALUŠÍK, Peter. *RAW image debayerization using deep neural network*. Brno: Brno University of Technology, Faculty of Electrical Engineering and Communication, Department of Telecommunications, 2023, 54 p. Bachelor's Thesis. Advised by prof. Mgr. Pavel Rajmic, Ph.D.

Author's Declaration

Author:	Peter Balušík
Author's ID:	230531
Paper type:	Bachelor's Thesis
Academic year:	2022/23
Topic:	RAW image debayerization using deep neural network

*The author signs only in the printed version.

ACKNOWLEDGEMENT

I would like to thank the supervisor of this thesis, prof. Mgr. Pavel Rajmic Ph.D, for consultations, valuable feedback and great suggestions involving this thesis. The GPU used for the research in this thesis was donated by the NVIDIA Corporation.

Contents

List of Figures

List of Tables

Introduction

Virtually every person has taken a photo in their life, whether with a digital camera or a phone. Only a few people know how that photo becomes colored. Briefly, a digital photo comes into existence by light passing through the lens of a camera and projecting itself onto an image sensor with a color filter. The color filter determines a specific color of the pixel (red, green, or blue) and the image sensor captures its value. Together they create a color filter array (CFA) sensor.

A CFA is designed with the idea that every array cell allows solely the red, green, or blue light component to pass. After the CFA sensor captures the individual colors a process called *demosaicing* begins. To put it briefly, demosaicing is a process of estimating the missing color values of a CFA. If the photo was taken out straight from the CFA sensor (before demosaicing), a RAW image would be obtained. Some people (usually photographers) choose this method to do the demosaicing process themselves – with an appropriate demosaicing software that is superior to the one in their camera.

Many demosaicing algorithms exist. The simplest one is the bilinear interpolation. Other, more effective, methods include more refined techniques – Malvar's [1] and Menon's [2] demosaicing method. This thesis tries to implement a completely different approach to demosaicing and that is using a deep convolutional neural network – with a revolutionary concept called Deep Image Prior [3].

This thesis consists of four main chapters. First chapter "Artificial neural networks" describes what are artificial neural networks and how they learn. It also described convolutional neural networks and some of their architectures. Second chapter "Deep Image Prior" briefly describes what is Deep Image Prior and the method it uses for image restoration. Furthermore, it focuses on the individual applications of DIP proposed in [3]. Third chapter "Demosaicing of RAW images" describes the Bayer filter, what are RAW images and several of the demosaicing methods. The last chapter "Experiments and result" describes the implementation of the proposed demosaicing method. It compares the proposed method (numerically, visually and in regard to computational difficulty) with other common demosaicing methods such as the ones mentioned above. Additionally, it describes the positives and negatives of the proposed method.

1 Artificial neural networks

An artificial neural network (ANN) is a system that is inspired by the human brain. An ANN is composed of multiple layers – the input layer, the hidden layers and the output layer. Each of these layers is made from nodes called neurons (also called perceptrons). Every neuron is a function that is given some input numbers and outputs a single number. For example, in a grayscale image, the input numbers would correspond to the grayscale values of pixels (a range from 0 for black pixels to 1 for white pixels). In the study of artificial intelligence, the output of a neuron is called the *activation* of a neuron. Each activation of neurons in one layer affects neurons in the other layer.

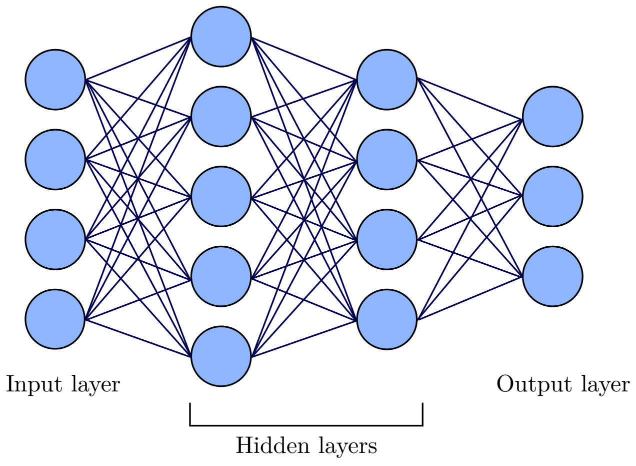


Fig. 1.1: A simple neural network with four inputs and three outputs. Blue circles represent neurons. The lines between individual neurons amount to the weights of an artificial neural network.

Different parameters are used to determine the activation of a neuron in the output and hidden layers. Individual layers of neurons are connected by *weights*. Weights are real numbers that can be adjusted, either by a human or the network itself, to influence the activation of neurons. To determine the activation in non-input layers, a weighted sum is computed from the weights and previous layer activations. In addition a *bias* is added to the weighted sum. The bias informs the network about how high the weighted sum needs to be before the activation starts having impact. Additionally, an *activation function* is used to compute the overall activation of a neuron. This is done for every neuron in every layer starting with the first hidden layer and ending with the output layer. The activation of neurons in the input layer is given by the input of an artificial neural network. The process of moving in only one direction, from the input layer to the output layer, is called forward propagation.

In earlier stages of neural networks a sigmoid function was used as the activation function. Nowadays, the sigmoid function has been replaced by the rectified linear unit or ReLU function. [4, 5]

A neuron can be described as the smallest computational unit of neural networks and is given by:

$$y = f\left(\sum_{i=1}^{D} w_i x_i + b\right) \tag{1.1}$$

where D is the number of neuron inputs (given by the number of weights), $\boldsymbol{x}$ is the input vector of neural activations, $\boldsymbol{w}$ is a set of weights corresponding to the input vector, b is the bias and f is a non-linear activation function (e.g. ReLU). Weights and biases of a neural network are sometimes combined into one parameter θ.

1.1 Deep learning

Deep learning is a learning technique mainly used for identifying objects in images, matching correct labels to images, speech recognition, language translation, search engines and data analysis. In deep learning the individual weights and biases are adjusted by the network itself. [4]

To learn, neural networks use a cost function, a data set, gradient descent and an algorithm called backpropagation (see Section 1.2). In the cost function a *cost* is calculated for each training example from the given data set. The cost can be calculated using variety of methods. For example, it can be calculated by adding up the square values of differences between the undesired outputs and the desired outputs. This sum is small if the outputs are the desired ones and large if the output values are wrong. The overall cost is usually calculated by averaging the cost for each training example. The cost function takes all the weights and biases of an ANN as an input and outputs the overall cost. The overall cost determines if a neural network operates correctly or the parameters of the neural network need to change. To minimize the cost function a gradient descent (see Section 1.2) is used. [5, 6]

1.1.1 Supervised learning

In supervised learning, the neural network is trained on a labeled data set. The data set is usually labeled by humans. A label is a known value specified on each example of the training data set. It could be as simple as a binary number, a category or a score. The learning algorithm is given an input–output training sample. It uses the knowledge of the desired output to adjust the parameters of a network accordingly. After a learning period the network is given new input vectors. The goal is to predict the output of these input vectors correctly.

Supervised learning can be divided into two main subcategories. First category being *classification*, which uses an algorithm to assign test data into particular categories. For example, in an image data set of animals, it divides them into mammals, reptiles, fish and so on. This process is done by drawing some conclusions about labeling of the specific entities. Common classification algorithms include linear classifiers, random decision forests, K-nearest neighbor and support vector machines. Second subcategory of supervised learning is *regression.* Regression models are used to understand the relationship between the input and the output. For example, it can be used for analyzing survey data, forecasting sales based on what the customers bought previously, or forecasting weather. Most common regression algorithms are linear regression and logistic regression,

Labeling large volumes of data is almost impossible. Also, labeling is not always as trivial as it sounds, not everything has a distinctive label. For that reason, semi-supervised learning exists. Semi-supervised learning is a middle ground between the supervised and unsupervised learning models, where a portion of the data set is labeled and the remaining data is unlabeled. It is most commonly used in extremely large data sets or when the extraction of certain features within the data set is difficult. In most cases, a small amount of labeled data can lead to significant improvements in time spent training and in accuracy of the neural network. [7, 8]

1.1.2 Unsupervised learning

In unsupervised learning, a neural network is trained without any labeled data. It is used to discover hidden patterns, regularities in data or cluster unlabeled data. It does not require any human supervision. The learning algorithm tries to label the data based on the features of the given input data. The ability to distinguish certain features is widely used in data analysis, image recognition or customer segmentation.

Unsupervised learning algorithms are used for three main tasks – clustering, association and dimensionality reduction. In *clustering*, similar data is grouped together. It is commonly used in customer segmentation, where customers are grouped together based on their common characteristics such as age or location. *Association* finds the relationships between variables in the data. For example, it is often used in market basket analysis, a technique to understand the customer purchasing patterns. In other words, which items are often bought together. Lastly, *dimensional reduction* is a technique which reduces the number of variables, while trying to preserve as much information about the input as possible. It is often used in the data preprocessing stage, where, e.g., the noise is removed from a corrupted image to improve the overall quality of the image. [9]

1.2 Backpropagation

A cost function (sometimes called loss function) is a multivariable function. A gradient is the direction of the steepest ascent. A negative gradient is the opposite of this gradient. To obtain the direction that decreases the cost function efficiently, a negative gradient vector is calculated. The negative gradient vector is calculated by computing the negative gradient of each variable. The negative gradient vector gives us the descent direction and additionally, the length of this vector indicates the steepness of the slope. Minimizing a cost function is just repeatedly calculating the negative gradient vector and taking a small step in that direction. This process is called gradient descent. Gradient descent takes the variables closer to a local minimum, where the output error is low on average. In other words, it minimizes the overall cost (output) of the cost function.

Backpropagation is the core algorithm behind how artificial neural networks learn. It computes the negative gradient vector. The magnitude of each negative gradient shows the sensitivity of the change. For example, if the negative gradient of one weight is 3 and the negative gradient of another weight is 0.2, the cost function is 15× more sensible to the change of the first weight. Backpropagation also determines how a single training example would change the weights and biases. It repeatedly computes negative gradients in each layer of a neural network starting at the output layer and outputs the desired change to the weights and biases. In other words, what changes to the parameters cause the biggest decrease of the cost function. A gradient descent step involves using backpropagation for all the training examples and averaging the desired changes (their negative gradients). This process is computationally very slow.

In common practice, a process called stochastic gradient descent is used instead of normal gradient descent. It consists of feeding the neural network with few random examples, computing the costs of the inputs, then averaging the negative gradients of the examples and adjusting the weights accordingly. This process is repeated for multiple small sets of examples in the training data set until the output of the cost function stops decreasing. [4, 6]

1.3 Convolutional neural networks

Convolutional neural networks (CNNs or ConvNets) process data from 1D, 2D and 3D arrays. For example, 1D arrays represent signals and sequences such as language, 2D arrays are used for the representation of audio spectrograms and images, and 3D arrays for videos and volumetric images. ConvNets are made form three types of layers – convolution, pooling, and fully connected layers. A typical CNN architecture consists of multiple convolution and pooling layers stacked on top of each other, followed by a few fully connected layers. These layers will be described for 2D arrays such as images. [4]

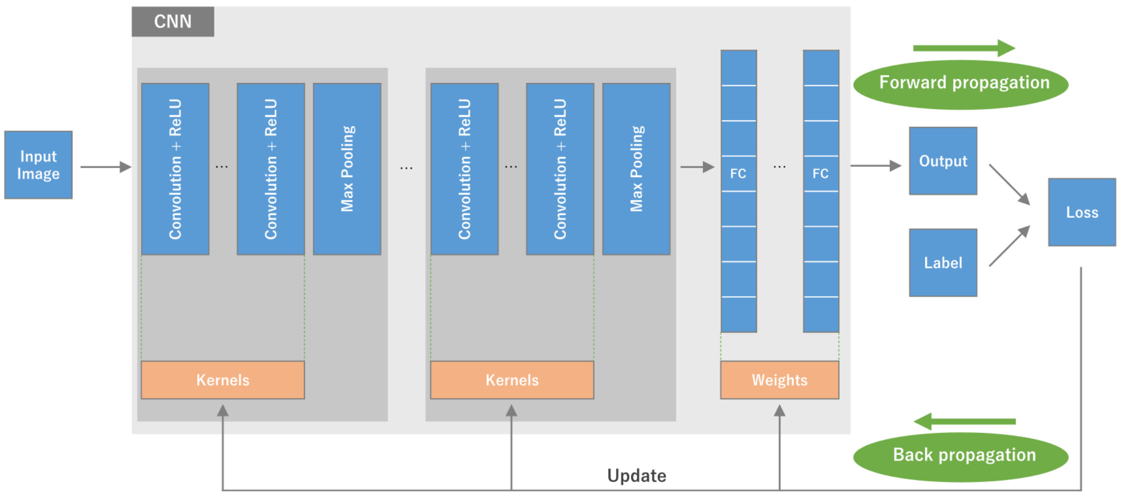


Fig. 1.2: A CNN architecture and its training process. Source: Adapted from [10].

The convolution layer is the key component of CNNs. It performs feature extraction, using operations such as convolution and the activation function. Convolution is a linear operation commonly used for detecting edges and other features. This detection is done using an array of small numbers called a kernel or a filter. The filter is applied across the input image, also called the input tensor. A Hamadard product (also called element-wise product) of each element of the filter and the input tensor is calculated in each position of the input image. These products are then summed and correspond to the output value in an output tensor called a feature map (see Fig. 1.3). The process of applying different filters is repeated to obtain multiple feature maps that represent different characteristics of the input image. Convolution uses two main parameters – the size and the number of filters (kernels). The most common filter size is 3×3 but sizes 5×5 and 7×7 are used occasionally. The number of filters is arbitrary. Sometimes a *padding*, usually zero padding, is used to prevent the reduction of dimensions of the feature map. It adds rows and columns of zeros to the input tensor, so the filter can be applied for places such as the outer bounds of an image. This allows the feature map to keep the same dimension throughout the convolution. Other parameter used in convolution is called a *stride*. A stride is the distance between two filters of an image. The most common stride is 1, that means the filter will move one pixel at a time. However, the stride can be bigger than one to achieve downsampling of the feature maps in the pooling layer. The outputs of the convolution are then passed through a nonlinear activation function, usually the ReLU. [10]

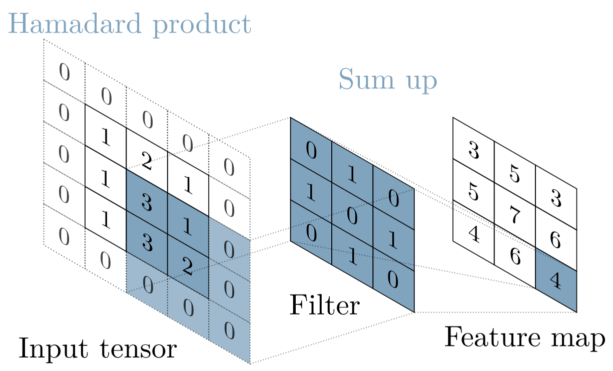


Fig. 1.3: An example of convolution with zero padding. The filter size and the stride are set to 3×3 and 1.

A pooling layer is used to reduce the in-plane dimensions of feature maps using a downsampling operation. It provides *translation invariance* to small shifts and distortions, and decreases the number of learnable parameters. Translation invariance means that the neural network produces the same output, regardless of the shift of the input. Two main pooling operations are discussed – max pooling and global average pooling. Max pooling is the most popular pooling operation. It divides the feature maps to multiple segments and outputs the maximum value of each segment. Other values are then discarded. Most common max pooling filter is a 2×2 filter with a stride of 2, which downsamples the feature map by a factor 2 (see Fig. 1.4). In global average pooling, a feature map is downsampled into a 1×1 array (1D array) by averaging all values in each feature map. This operation is usually applied once before the fully connected (FC) layers. Additionally, it can be used to replace the fully connected layers. It also reduces the number of learnable parameters and accepts inputs of variable sizes.

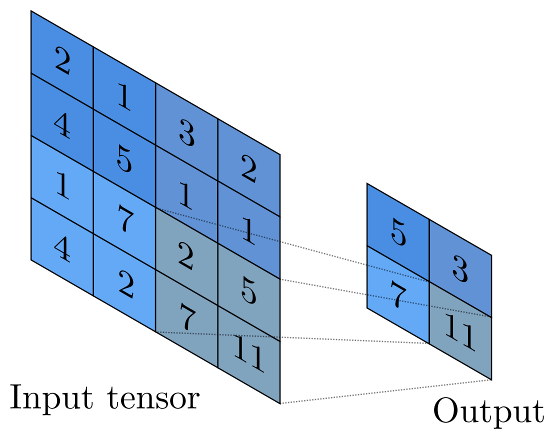


Fig. 1.4: An example of max pooling with no padding. The filter size and the stride are set to 2×2 and 2.

A fully connected layer is a layer, where every input is connected to every output by a learnable weight. The output feature maps of the last convolution or pooling layer are transformed into an 1D array and connected to one or more fully connected layers. In classification tasks, the final fully connected layer has the same number of outputs as is the number of classes. Each fully connected layer is followed by an activation function like ReLU. The final fully connected layer can be followed by a softmax function. Softmax function converts an output vector of k real numbers into a probability distribution of k possible outcomes. [10]

1.3.1 Architectures

A networks architecture is a crucial factor in improving the performance of different applications. Because of that many CNN architectures exist. Some examples include: AlexNet, Network-in-network, VGG, GoogLeNet, ResNet, DenseNet, CapsNet, U-Net and many others. AlexNet and U-net are briefly described below.

AlexNet [11] is a highly respected deep CNN architecture, that began the research era in CNN applications. It was made in 2012 and excelled in many fields of image recognition and classification. AlexNet improved the learning ability of neural networks by increasing their depth. It also implemented multiple parameter optimization methods. Two graphics processing units (GPUs) were used in parallel to train the first AlexNet to overcome the limited learning ability of 2012 hardware.

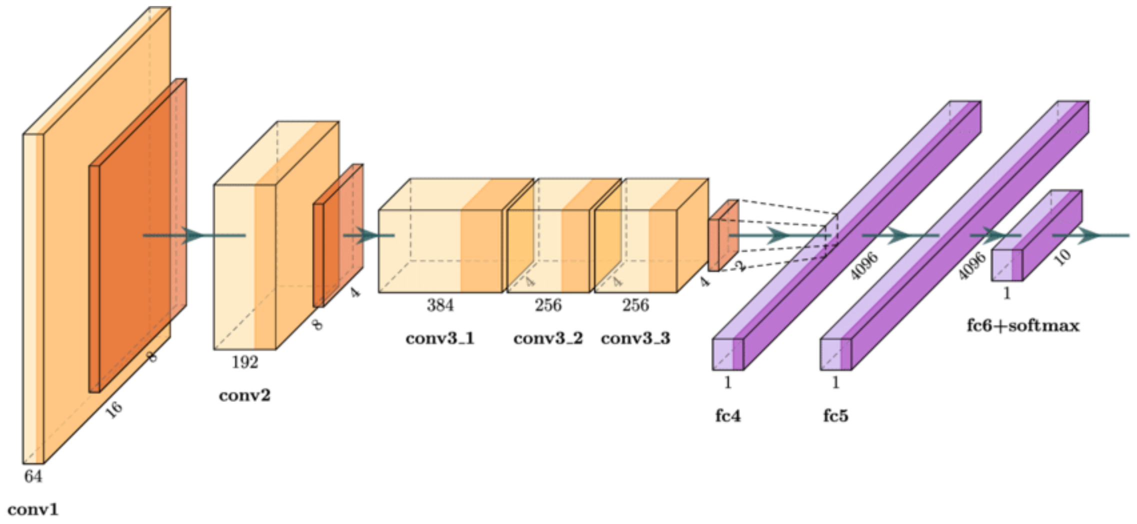


Fig. 1.5: A modern AlexNet architecture. The convolution layer is represented by light (convolution) and slightly darker orange boxes (ReLU funtion). Darker orange boxes represent the max pooling layers. Light and darker purple (ReLU function) boxes indicate the individual fully connected layers. Source: Adapted from [12].

The depth of this CNN is eight. The first five layers are convolutional and other three are fully connected layers. The output of the last fully connected layer is given to a softmax function. After every convolutional layer it uses special max pooling layers called overlapping pooling layers described in [11]. A modern AlexNet architecture can be seen in Figure 1.5.

U-net [14] is a modification of the fully convolutional network proposed in [13]. It is modified to work with fewer training images and allows for more precise segmentation. The main idea is to supplement the network with layers where pooling operators are replaced with upsampling operators. These layers increase the resolution of their outputs. Therefore, the following convolutional layers give more precise outputs.

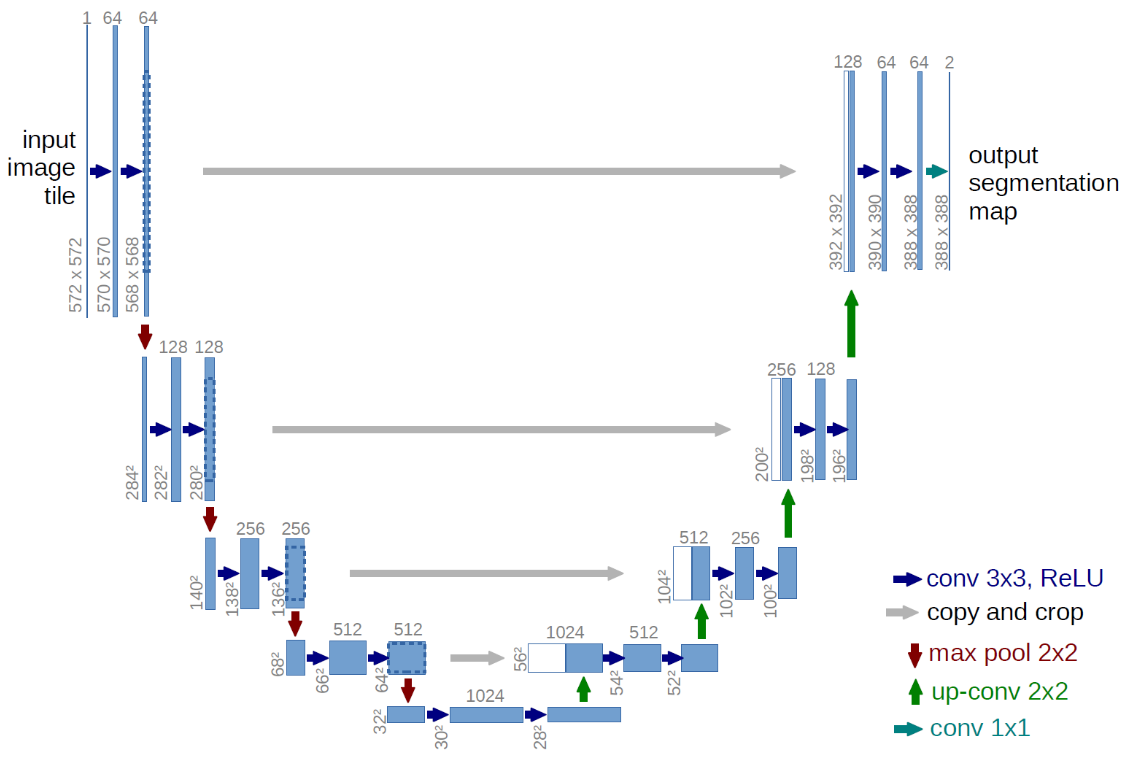


Fig. 1.6: The U-net architecture. Each blue box represents a feature map. The number of feature channels is written on top of the box. The size of the layer can be seen in the bottom left corner of the box. White boxes constitute copied feature maps. Gray arrows represent skip connection and other arrows represent different operations. Source: Adapted from [14].

The upsampling layers provide more feature channels, allowing the network to propagate context to higher resolution layers, which results in the u-shaped architecture (Fig. 1.6). This process is done with large *skip connections*. Skip connections, as the name suggests, skip some layers in neural networks and feed the output of one layer as the input in other layers. The architecture does not contain any fully connected layers. It uses an overlap-tile strategy that allows for flawless segmentation of large images. This strategy is described in [14]. The architecture consists of two paths. First being a contracting path (encoder path), that follows the typical CNN architecture. The second being an expansive path (decoder path), where the feature maps are upsampled. At the final layer an 1D array convolution (1×1 convolution) is used to map each feature vector to the number of classes. The network consists of 23 convolutional layers. The most common application of U-net architecture is in biomedical image segmentation. [14, 15]

2 Deep Image Prior

Deep Image Prior (DIP) [3] is a revolutionary concept that uses a generative convolutional neural network (CNN). It is used for image reconstruction problems like denoising, super-resolution, inpainting and others. Generally such problems are solved by CNNs *trained* on large data sets. This remarkable concept uses a CNN that is completely *untrained.* The only input information of the CNN is an image damaged or degraded in some way and the structure of the network. All the weights of the network are randomly initialized and serve as a parameterization of the image. DIP proves that the architecture of a neural network affects the results.

This neural network connects the gap between two most popular image restoration methods. The first method uses learning-based convolutional neural networks and the other is non-trained, purely based on handcrafted natural image priors such as Total Variation (TV). DIP uses an U-Net-like "hourglass" architecture also called encoder–decoder with skip connections. It demonstrates that an untrained deep convolutional generator can replace the natural prior used in TV with significantly better results. [3]

2.1 Selected applications of DIP

Before talking about the applications, the **method** used in DIP needs to be understood. A deep neural network can be considered as a parametric function $x = f_\theta(z)$ that maps a code vector z to an image x. The code vector z is fixed and randomly initialized. This method displays that a notable amount of information is stored just in the structure of the function f_θ. The parameter θ consists of the weights and the bias of the filters in the network. Operations like the linear convolution, upsampling and non-linear activation functions are used by the network. [3]

In reconstruction tasks, a restored image can be obtained by maximizing posterior distribution. Conditional (posterior) distribution can be defined as $p(x|x_0)$, where an image x is made from corrupted counterpart x_0. Instead of working with the distribution explicitly, the task is interpreted as energy minimization:

$$x^* = \min_x E(x; x_0) + R(x), \tag{2.1}$$

where $E(x; x_0)$ is the data term that is chosen by the given application, x_0 is the damaged image and $R(x)$ is an explicit regularizer. The regularizer $R(x)$ can be removed since it is not usually tied to particular applications. As its alternative, an implicit *prior* captured by the parameterization of a neural network is used:

$$\theta^* = \operatorname*{argmin}_\theta E(f_\theta(z); x_0), \tag{2.2}$$

where the minimizer θ^* is found using gradient descent. The applications start from a randomly initialized parameter θ, that is why the only input information needed is the corrupted image x_0. The result, given the minimizer θ^*, is obtained as $x^* = f_{\theta^*}(z)$. An important point to mention is that the optimization process needs to be stopped after a specific number of iterations because the result image x^* will eventually turn (in some applications) into the input image x_0. Deeper dive into this method is shown in the journal [3].

2.1.1 Denoising

Deep image prior can be used to remove noise from an image. Denoising generates a clean image x from a noisy equivalent x_0. There are two types of noise. First type is when the noise model is known, usually as: $x = x + \epsilon$, where ϵ follows a specific probability distribution. Second type is when the noise model is not defined.

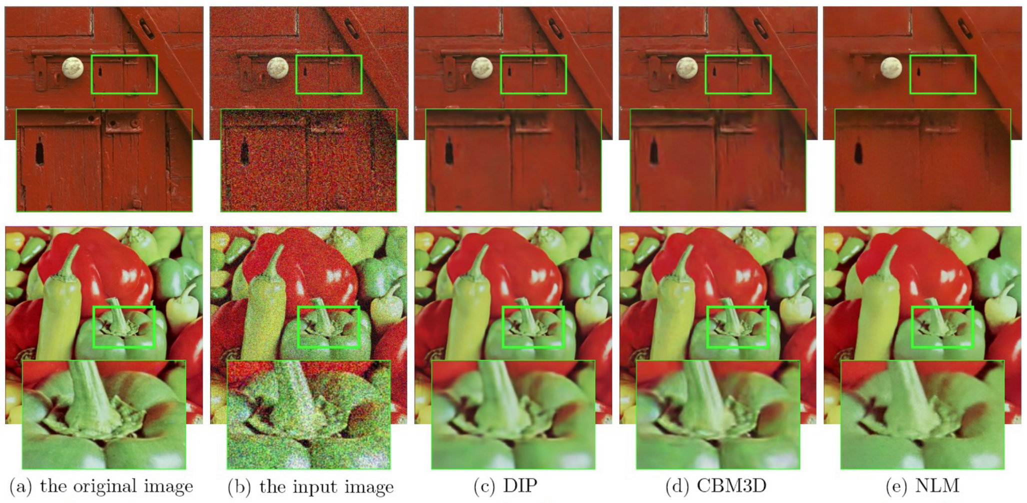


Fig. 2.1: Deep image prior denoising comparison with the CMB3D and NLM. Source: Adapted from [3].

DIP works with the second type, but it can be replaced with the first type. The equations for this problem are as follows:

If the value of the minimizer θ^* needs to be found, it can be done by optimizing (2.2) using data term that compares the image x and the noisy image x_0

$$E(x; x_0) = ||x - x_0||^2. \tag{2.3}$$

If it is plugged to equation (2.2), it leads to the optimization problem

$$\min_{\theta} ||f_\theta(z) - x_0||^2. \tag{2.4}$$

After replacing the minimizer θ^* of (2.2), a clean image $x^* = f_\theta^*(z)$ is given.

This approach can also be used for restoration of JPEG-compressed images. While the restoration process is running, it can remove a great deal of JPEG-compression artifacts after about 2400 iterations, but ultimately becomes the input image at 50K iterations.

The peak signal-to-noise ratio (PSNR) of this method is 29.22 after 1800 iterations. If the restored images in the last iterations are averaged, the PSNR of about 30.43 is acquired. Maximal score of this method is 31.00 PSNR. It performs greatly compared to other popular methods like CMB3D [16] and Non-local means (NLM) [17] that achieved 31.42 and 30.26 PSNR. The comparison of these methods can be seen in Figure 2.1. [3]

2.1.2 Super-resolution

Super-resolution takes a low resolution (LR) image $x_0 \in \mathbb{R}^{3\times H\times W}$, upsampling factor t and it produces an upsampled image $x \in \mathbb{R}^{3\times tH\times tW}$ also called high resolution (HR) image. To find a solution to this inverse problem, the data term in (2.2) is set to:

$$E(x; x_0) = ||d(x) - x_0||^2, \tag{2.5}$$

where $d(.) : \mathbb{R}^{3\times tH\times tW} \to \mathbb{R}^{3\times H\times W}$ is a *downsampling operator* that rescales the image by upsampling factor t. The problem is finding a HR image x that is after downsampling identical to LR image x_0. There is infinite number of solutions for

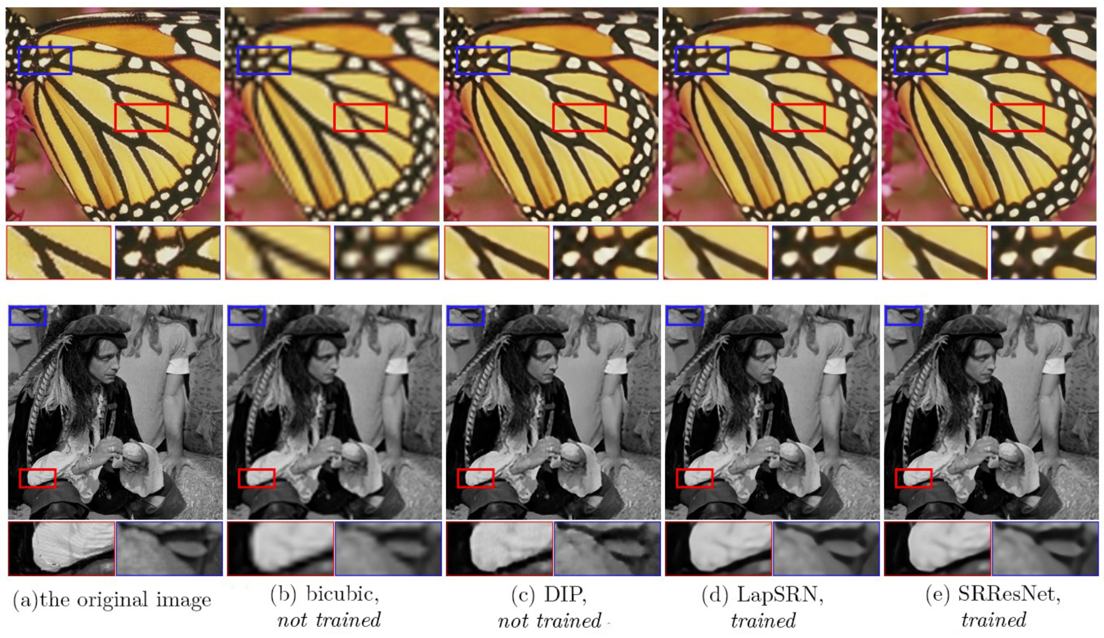


Fig. 2.2: Deep image prior 4× super-resolution comparison with the bicubic upsampling, LapSRN and SRResNet. Source: Adapted from [3].

the ill-posed problem that is super-resolution. As a result, regularization is needed to choose the most reasonable from the infinite number of minimizers of (2.5). Regularization is done by reparameterization $x = f_\theta(z)$ and optimization of the result with respect to θ. Neural networks and downsampling operators like Lanczos are differentiable, so gradient descent is used for optimization. [3]

Tab. 2.1: Detailed 4× super-resolution PSNR comparison on the data set from [20].

4× super-res	Baby	Bird	Butterfly	Head	Woman	**Avg.**
No prior	30.16	27.67	19.82	29.98	25.18	26.56
Bicubic	31.78	30.20	22.13	31.34	26.75	28.44
TV prior	31.21	30.43	24.38	31.34	26.93	28.85
Glasner et al.	**32.24**	31.10	22.36	**31.69**	26.85	28.84
DIP	31.49	**31.80**	**26.23**	31.04	**28.93**	**29.89**
LapSRN	33.55	33.76	27.28	32.62	30.72	31.58
SRResNet-MSE	33.66	35.10	28.41	32.73	30.60	32.10

Tab. 2.2: Detailed 8× super-resolution PSNR comparison on the data set from [20].

8× super-res	Baby	Bird	Butterfly	Head	Woman	**Avg.**
No prior	26.28	24.03	17.64	27.94	21.37	23.45
Bicubic	27.28	25.28	17.74	28.82	22.74	24.37
TV prior	27.93	25.82	18.40	28.87	23.36	24.87
SelfExSR	**28.45**	26.48	18.80	29.36	24.05	25.42
DIP	28.28	**27.09**	**20.02**	**29.55**	**24.50**	**25.88**
LapSRN	28.88	27.10	19.97	29.76	24.79	26.10

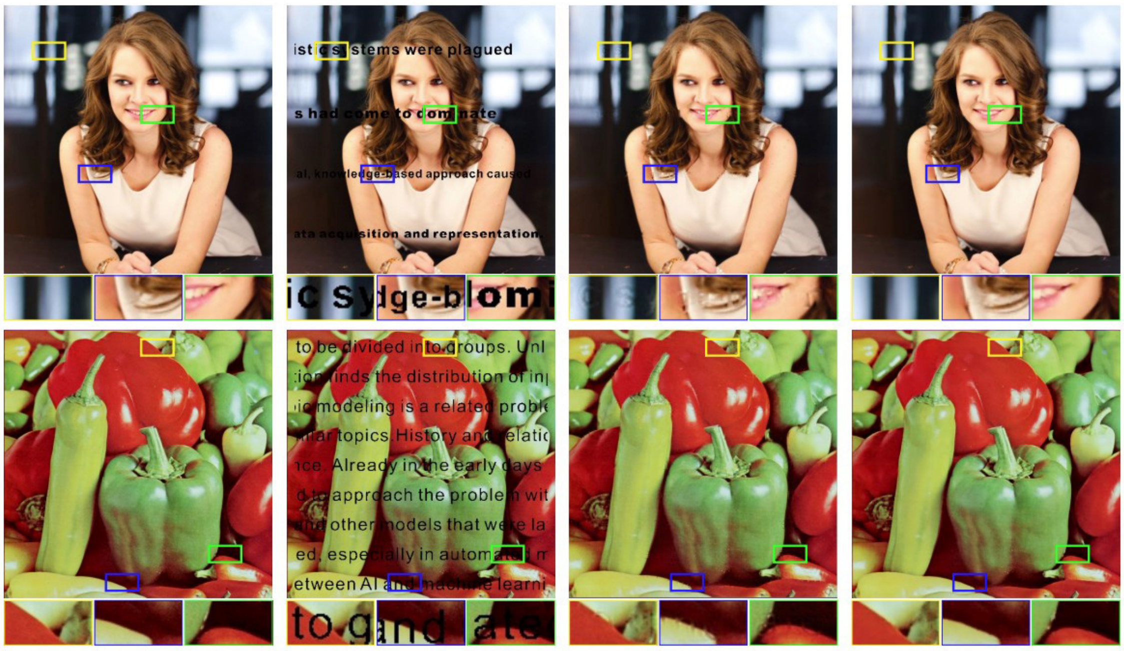
Next, the quality of this method is compared to other methods, such as bicubic upsampling, modern learning-based methods like SRResNet [18], LapSRN [19] and others. For this comparison the numbers 4 and 8 are chosen to be the factor t. Furthermore, the peak signal-to-noise ratios (PSNRs) are calculated. Visual comparison can be seen in Figure 2.2 and computed PSNRs in Tables 2.1.2 and 2.1.2 adapted from [3]. Visually, this method reaches the quality of learning-based methods. By comparing the PSNRs it can be seen that this method is surpassed by the modern learning-based methods, but still produces better results then bicubic upsampling and other non-trained methods. [3]

2.1.3 Inpainting

Inpainting is a process where damaged, destroyed or missing portions of an image are repaired (repainted). In this case, a portion of pixels is missing. A damaged image x_0 with binary mask $m \in \{0,1\}^{H\times W}$ of missing pixels can be restored with data term

$$E(x; x_0) = ||(x - x_0) \odot m||^2, \tag{2.6}$$

where $\odot$ is the Hadamard product. Data prior is needed, due to separation of the energy and the missing sections. If the data prior was not present, the image would not change after optimization over pixel values x. The prior is given by optimization of the data term with respect to the reparameterization in (2.2).



(a) the original image (b) the corrupted image (c) Shepard networks (d) DIP

Fig. 2.3: Deep image prior text inpainting comparison with Shepard networks. Source: Adapted from [3].

Inpainting can be used for tasks such as removing text, filling missing segments of pixels or restoring image with a certain percentage of pixels missing. This method does not work for more semantic inpaintings like human or animal faces, equations, text and others, due to not being trained. However, for other situations it works wonderfully. For removing text it works better in comparison with other methods like the Shepard networks method [21], which can be seen in Figure 2.3. When a certain percentage of pixels is missing, where the mask is randomly sampled according to a binary Bernoulli distribution, it performs greatly, as opposed to convolutional sparse coding (CSC) [22]. Example of 50 % of pixels missing is shown in Figure 2.4. In architectural comparison, having deeper architectures proved to be favourable. Also having skip-connections was shown to work greatly for recognition tasks. [3]

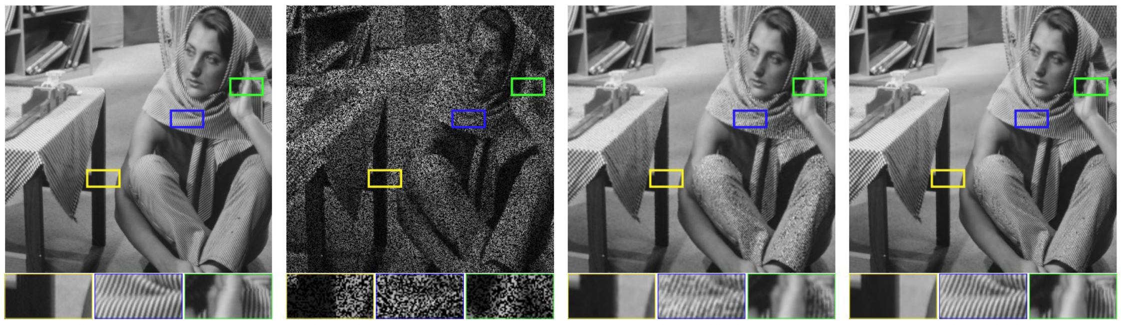


Fig. 2.4: Deep image prior 50 % of pixels missing comparison with CSC. Source: Adapted from [3].

2.1.4 Other applications

Other applications presented in [3] include flash – no flash reconstruction, activation maximization and image enhancement.

Flash – no flash reconstruction differs from other methods by its need of two input images. One shot with camera flash which procures sharpened details, the other shot with natural lighting. The goal of this application is to combine these images into one having both sharp details and natural colors. Good reconstructions were acquired by using (2.4) for denoising, with image shot with camera flash as in input z. The result is given by guided image filtering.

Activation maximization is a method to visualize inner parts of a deep neural network. It synthesizes an image where a specific neuron is active by solving the optimization problem:

$$x^* = \arg\max_x \phi(x)_m, \tag{2.7}$$

where m is the index of a neuron. The layers of AlexNet and VGG-16 CNNs are used for this application.

Image enhancement is used for high frequency enhancement of an image. It is done by using (2.4) and choosing the target image to be x_0. By stopping the optimization process after particular number of iterations, image x_c is obtained. Detailed image (x_f) is calculated by subtracting the image x_c from the target image x_0. The enhanced image (x_e) is then constructed with equation: $x_\mathrm{e} = x_0 + x_\mathrm{f}$.

Similar to denoising, undesirable high frequency details start to appear after a certain amount of iterations. [3]

3 Demosaicing of RAW images

Demosaicing, also called color filter array (CFA) interpolation, is a process of estimating missing color values from a CFA. Multiple color filter array patterns exist. Bayer filter is one of the most popular CFA patterns. (Fig. 3.1).

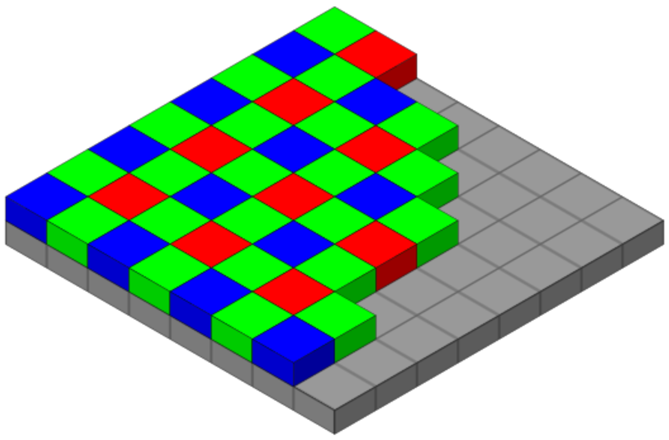


Fig. 3.1: The Bayer pattern. Source: Adapted from [23].

To acquire an RGB color of an image *precisely* a digital camera would require three color sensors. Each sensor would need proper driving electronics and precise placement. Therefore, measuring the precise color could quickly get expensive. For this reason many cameras use a single CFA sensor, allowing them to measure only single color value of each pixel. The other values are then obtained by demosaicing. The gray squares in Fig. 3.1 represent the image sensor. These squares combined with the colored squares make up a CFA sensor. [24]

3.1 RAW image

A digital camera is made of an optical system, an image sensor and an electronic system. The output signal of an image sensor is analog. This signal is then converted by an A/D converter into a digital signal, that is recorded as digital image in RAW format. RAW image is uncompressed and contains all the original information about the image. The image sensor is not capable of separating the different color channels, thus a CFA sensor like the Bayer pattern sensor is necessary. [25]

RAW files are produced not only by digital cameras, but also image scanners and film scanners. They are created in multiple kinds of proprietary file formats. Like .nef for Nikon cameras, .cr2 or .crw for Canon, .arw file used by Sony cameras and many others. These files can be viewed, edited and converted into other formats in multiple programs for image editing like Adobe Photoshop, Adobe Lightroom or Zoner Photo Studio. [26]

RAW image files have the best detail possible compared to other raster or vector files formats. They contain significantly more colors compared to a format like JPEG. RAW files can be converted to lossless compressed RAW files that occupy less space. Additionally, there are many adjustments to be made without changing the RAW file itself. Meaning all the changes are never permanently applied to the RAW file, instead they are saved in the editing software. Some adjustments include: changing the image from grayscale to an RGB image, changing contrast, brightness and others. A problem may arise in regards to the size, as it is larger compared to compressed images like JPEG, or with certain software that cannot read formats from particular cameras. Many people do not have the appropriate software to open them; hence, they are more difficult to work with and share. [27]

3.2 Bayer filter

Bayer filter is the most common color filter array (CFA) used for demosaicing. It is named after Bruce Bayer who invented it in 1974. CFA is a mosaic like pattern put over the pixel sensors of an image. The Bayer filter is made from red (R), green (G) and blue (B) colors that are arranged in a square pattern and displayed over the whole filter. There are four types of this square pattern RG–GB, BG–GR, GR–BG and GB–RG as shown in Fig. 3.2. [24]

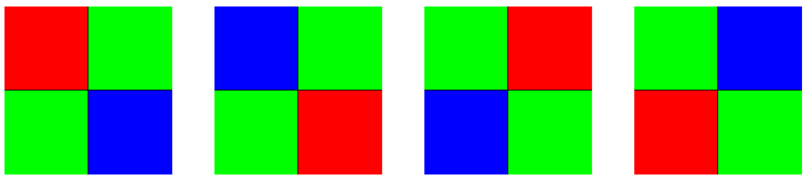


Fig. 3.2: Types of Bayer patterns.

The pattern uses two green, one red and one blue component. Therefore, in a 64 Megapixel Bayer pattern camera sensor, 16 Mpx store the red values, 16 Mpx store the blue values and 32 Mpx store the green values. The higher focus of Mpx is attributed to the heightened sensitivity of human eyes to the color green.

3.3 Methods of demosaicing

As mentioned, demosaicing is a process of calculating the missing color values from a color filter array. Nowadays, a large amount of demosaicing methods exists. The simplest method of demosaicing is bilinear interpolation. In this method the three color channels are independently interpolated using symmetric bilinear interpolation from the neighbors of the same color. Bilinear interpolation does not account for the correlation between color values. Due to this it creates significant artifacts, mainly near edges and other high-frequency content. [1]

For the explanation of the bilinear interpolation, the Bayer pattern shown in Figure 3.3 is used.

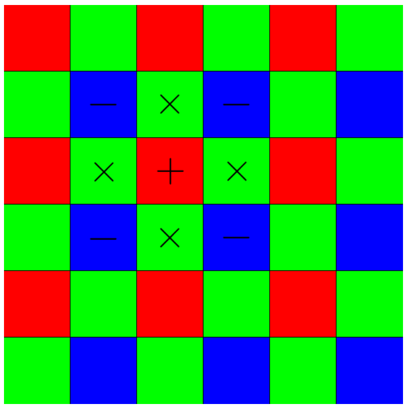


Fig. 3.3: The Bayer pattern for demonstration.

To obtain the green value $g(i, j)$ at position (i, j) (marked with '+' in Fig. 3.3) of a red or a blue pixel, the average of surrounding green pixels (marked with '×' in Figure 3.3) is calculated:

$$\hat{g}(i, j) = \frac{1}{4} \sum g(i + m, j + n) \tag{3.1}$$

where $(m, n) = \{(0, -1), (0, 1), (-1, 0), (1, 0)\}$. In Figure 3.3 this equation can be used when calculating the green value of the red pixel marked with '+' sign. For the red or blue values, in this case blue values marked with '−', the same equation applies with the difference that $(m, n) = \{(-1, -1), (-1, 1), (1, -1), (1, 1)\}$. This approach works for red and blue pixel positions. To calculate the R and B value in the position of a G pixel, only the two R and B neighbouring values are averaged. At the borders of an image, only values inside the bounds of the image are used for the calculations. No overflow logic is needed, because the output value has the same dynamic range as the input value. Meaning that the output value cannot move outside the range 0–255 because bilinear interpolation is calculated by averaging specific values in the same range. [1]

Demosaicing methods can be divided into multiple groups; however, only the major two are described. The first group includes heuristic approaches. The second group interprets demosaicing as an image restoration problem.

3.3.1 Heuristic approaches

Heuristic approaches, often called heuristic methods, are primarily filtering operations based on logical assumptions about an image. In other words, they do not correspond to mathematically defined optimization problems. Already mentioned bilinear interpolation is a heuristic approach that ignores to the correlation between color channels. Other heuristic methods may utilize this correlation for a better result. Several of these approaches exist, a portion of them is described below.

Edge-directed interpolation
Nonadaptive methods like bilinear interpolation and bicubic interpolation, which are very similar with the main difference being the number of pixels interpolated through, generally fail near edges and other highly-textured regions. On the contrary, edge-directed interpolation is an adaptive method. In this method, each neighboring pixel in a 3×3 region is analyzed. The favourable interpolation direction is then chosen to avoid interpolating across edges and other textured content.

The choice of an interpolating direction can be done using multiple strategies. For example, a red pixel in the Bayer pattern (Fig. 3.3) is taken. One strategy is that the horizontal ΔH and vertical ΔV gradients of surrounding green pixels are calculated, than compared. If $\Delta H > \Delta V$, the interpolation is done in the vertical direction. If $\Delta H < \Delta V$, the interpolation is done in the horizontal direction and if $\Delta H = \Delta V$, intuitively, the interpolation is done along both directions. Another strategy compares the gradients to a constant threshold, like method in [28]. If the gradient in one direction is below the selected threshold, the interpolation is done in that direction. When both gradients fall below or above the threshold, the missing value is obtained by interpolating along both directions. [24]

The edge-directed interpolation can also be altered by using bigger regions than 3×3. These modifications use more complex strategies to choose the interpolation direction and make full use of texture similarity in different color channels. Consequently, interpolation over e.g. 5×5 region performs better than the 3×3 counterpart.

Constant-hue-based interpolation
In demosaicing, there are many assumptions made about the correlation of colors. One common assumption is that the hue (pure color) of an object in an image is constant. Even though the brightness of an unicolored object changes, the hue is still constant, although the originally measured color values might change. Thence, an assumption that the color differences (distance between two colors) within objects are also constant is made. This prevents sudden changes in hue and has been used for interpolation of the red and blue channels by multiple algorithms such as [29].

Initially, the green channel is interpolated using bilinear or edge-directed interpolation. Additionally, the red hue (ratio between red and green channel) and the blue hue are interpolated. The values of the R and B channel are then estimated from the interpolated hues. This estimation is done by multiplying the particular hues with the green value. For the interpolation of hues, any interpolation method can be utilized. Similar approach can be used for constant-difference-based interpolation, only with color differences instead of hues.

Pattern matching
As the name suggests, this approach tries to find a specific pattern in the color values or match the values to established templates. Each template uses a different interpolation method. A pattern matching algorithm is described in [30]. It is used on the green image, where every missing green value classifies as an edge, corner, stripe or other feature usually found in natural images. Afterwards, a corresponding interpolation method is chosen to find the missing color values. [24]

3.3.2 Reconstruction approaches

Reconstruction methods, unlike heuristic methods, try to solve a mathematical problem based on various beliefs about the correlation between channels and the prior image. Several methods use iterative algorithms. Other interpret the problem as a Bayesian estimation problem, where the beliefs about hue and spatial smoothness are used for regularization. The Bayesian and artificial neural network approach will be described.

Bayesian approach
Bayesian approach utilizes the noise information and prior knowledge (such as constant hue, color difference, spatial smoothness) about the given image. It uses the maximum a posteriori probability (MAP) estimate. The variables such as the observed data $O(n_1, n_2)$, the color channels $S(n_1, n_2)$ and additive noise $N_S(n_1, n_2)$ are assumed to be random. The MAP estimate $\hat{S}$ is given by

$$\hat{S} = \arg\max_S \{p(S|O)\} = \arg\max_S \{p(O|S) \cdot p(S)\}, \tag{3.2}$$

where the $p(O|S)$ is a conditional probability density function (PDF), $p(S)$ is a prior PDF. The $p(O|S)$ is obtained from the noise model, usually an additive white Gaussian noise. The prior PDF ($p(S)$) is acquired using variety of methods. For example, using Gibbs distribution. Gibbs distribution (also called Boltzmann distribution) is described by an energy function $U(x)$ and a temperature parameter T. A PDF with this distribution can be written as

$$p(x) = \frac{1}{Z} \mathrm{e}^{-U(x)/T}, \tag{3.3}$$

where Z is the normalizing constant. In [31], the energy function is written as sum of the local energy functions at each pixel location. After defining the individual energy functions, the total energy can be minimized with consideration of two main factors. First, the image should be smooth i.e. the deviation of pixel values should be small. Second, the color cross-ratios around the given pixel location should be equivalent. The energy model is defined by the linear combination of these two factors. [24, 31]

Artificial neural network approach
This approach learns the parameters for reconstruction from a set of training images. It is used by Kapah and Hel-or in [32]. They propose three methods – perceptron, backpropagation and quadratic perceptron. The inputs are obtained from a 2×2 region. The pixels around this region count towards the inputs. Thence, the neural network uses 16 inputs and produces 8 outputs. Each method calculates the outputs (missing color values) differently. [24]

The perceptron method acquires them using linear combination of the inputs. It performs sufficiently in low-frequency regions, but its performance drops in high-frequency regions. The backpropagation method can learn complex nonlinear functions using sigmoid functions. Therefore, it produces better results in high-frequency regions, compared to low-frequency regions. This problem is solved by a selector that uses different methods depending on the selected 2×2 region. The last method is a quadratic perceptron network, where the weights are functions of inputs. They are obtained by a perceptron subnetwork. Deeper explanation of the quadratic perceptron method can be found in [32].

3.3.3 Malvar's method

Malvar's interpolation method is a heuristic method used in this thesis, due to its simple nature. It uses bilinear interpolation and a gradient correction to estimate the output color value more accurately. For example, if the green value of a red pixel location needs to be calculated, the red value at that location is not discarded. Instead, it is compared to the bilinear interpolation of nearest red samples. If it is different from the estimate of bilinear interpolation, there might be a sharp luminance change at that pixel. In other words, our pixel is probably near an edge; therefore, the bilinearly interpolated green value needs to be corrected by adding some amount of the luminance change. To interpolate green values at a red pixel position, this equation is used:

$$\hat{g}(i,j) = \hat{g}_{\mathrm{B}}(i,j) + \alpha \cdot \Delta_{\mathrm{R}}(i,j) \tag{3.4}$$

where subscript 'B' means bilinearly interpolated, α is a gain factor controlling the correction amount and $\Delta_{\mathrm{R}}(i,j)$ is the gradient of R at that position. It is given by:

$$\Delta_{\mathrm{R}}(i,j) \triangleq r(i,j) - \frac{1}{4}\sum r(i+m, j+n) \tag{3.5}$$

where $(m,n) = \{(0,-2),(0,2),(-2,0),(2,0)\}$. As can be seen, the bilinear interpolation is corrected with a gradient multiplied by a gain factor. To interpolate green value at blue positions, the same equation is used only with gradient $\Delta_\mathrm{B}(i,j)$. For interpolating red values at green pixels, this equation is used:

$$\hat{r}(i,j) = \hat{r}_\mathrm{B}(i,j) + \beta \cdot \Delta_\mathrm{G}(i,j) \tag{3.6}$$

where the gradient $\Delta_\mathrm{G}(i,j)$ is a 9-point region and $(m,n) = \{(0,-2),(0,2),(-2,0),(2,0),(-1,-1),(-1,1),(1,-1),(1,1)\}$. For the interpolation of red value at the position of a blue pixel, this equation is used:

$$\hat{r}(i,j) = \hat{r}_\mathrm{B}(i,j) + \gamma \cdot \Delta_\mathrm{B}(i,j) \tag{3.7}$$

where the gradient $\Delta_\mathrm{B}(i,j)$ is a 5-point region and (m,n) is the same as for the equation (3.5). For interpolating blue values the equations are similar by symmetry. The gradient 9-point and 5-point regions are shown in Fig. 3.4. [1]

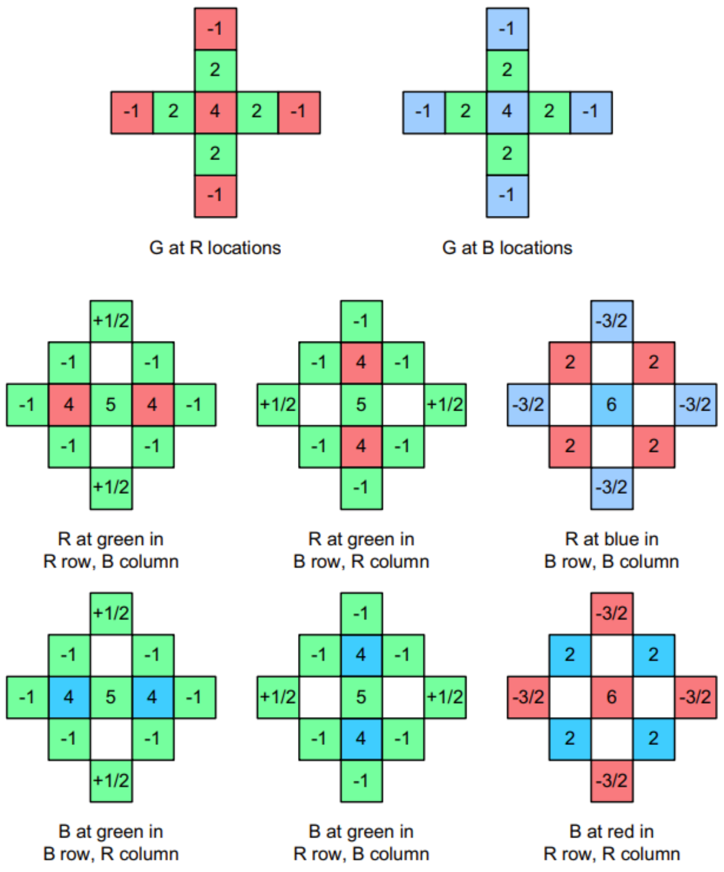

G at R locations
G at B locations
R at green in R row, B column
R at green in B row, R column
R at blue in B row, B column
B at green in B row, R column
B at green in R row, B column
B at red in R row, R column

Fig. 3.4: The optimal filters. Source: Adapted from [1].

To determine the gain factors $\{\alpha, \beta, \gamma\}$, the values that led to minimum mean-square error between the estimated value and the correct value were selected. These factors were computed from the Kodak image data set used in [33]. The result of this computation: $\alpha = 1/2$, $\beta = 5/8$ and $\gamma = 3/4$. An optimal linear FIR filter coefficients are determined by these factors. The optimal filters for a 5×5 region are shown in Fig. 3.4. This method corrects the bilinearly interpolated values by adding some amount of luminance change. In some cases, this correction is done using a large number, causing the output value to overflow the range 0–255. Therefore, this method needs the presence of some overflow logic. [1]

3.3.4 Menon's method

Menon's approach [2] is a bit different from standard methods. It employs directional interpolation of the green positions of pixels. This directional interpolation is done along both horizontal and vertical direction and creates two "green" images. After the interpolation, a decision is made between the two images. This decision is made using two classifiers that are described in [2]. Following the decision, the red and blue components are interpolated. Apart from other methods, where only the information captured by the Bayer filter is utilized, the two classifiers and the selected "green" image help with the calculation of these (blue, red) components. This approach results in more accurate images with less artifacts. [2]

4 Experiments and results

This chapter introduces the implementation of a new approach to demosaicing and that is with a relatively new concept called Deep Image Prior [3]. It also describes multiple experiments and comparisons done using this method. All the experiments used a traditional data set called a Kodak image data set [33]. All 24 images with the size of 768×512 or 512×768, depending on the image orientation, were used. The images came in an uncompressed PNG-24 file format with 8 bits per channel.

The proposed method was compared with four commonly used demosaicing methods – the bilinear interpolation (Section 3.3), Malvar's method (Section 3.3.3), Menon's method (Section 3.3.4) and the `demosaic` function in Matlab.

4.1 Demosaicing using Deep Image Prior

Demosaicing using the DIP works similarly to other applications proposed in [3], more precisely the application of inpainting. As mentioned in Section 2.1.3, a binary mask $m \in \{0,1\}^{H\times W}$ is used to represent the missing image pixels that need to be reconstructed. The data term (2.6) is used to restore the image. Considering the behavior of DIP, demosaicing can be thought of as a form of inpainting, since both of them try to find the missing values of pixels, only instead of the binary mask m, a new mask M_{CFA} is presented. The data term for demosaicing becomes

$$E(f_\theta(z); x_0) = ||(f_\theta(z) - x_0) \odot M_{\text{CFA}}||^2. \tag{4.1}$$

Similarly to inpainting, the inclusion of the data prior during the optimization of the data (energy) term is crucial.

4.1.1 Parameters of the neural network

Multiple architectures and their configurations were tested for the proposed demosaicing method. The most promising proved to be an encoder-decoder architecture similar to the U-net [14]. The configuration includes five downsampling and five upsampling convolutional layers. Unlike the U-net (Fig. 1.6), skip connections (even though they led to faster optimization) were not used due to the fact that they brought undesirable visual artifacts, at least with the configuration used in the experiments. Both upsampling and downsampling layers used the same number of feature maps, which were as follows: 64, 64, 128, 256, 512. This setting was the most distinctive compared with the inpainting experiment in [3]. A 3×3 convolution with zero padding was used to obtain the feature maps. This convolution was followed by a LeakyReLU activation function. The weights of the neural network were randomly initialized with Gaussian noise and the chosen learning rate was 0.001. As for the optimization algorithm, the Adam optimizer [34] was used. The highest PSNR values frequently appeared around 2000 epochs (see Fig. 4.1); therefore, 2000 epochs were chosen to be the stopping point of learning. The SSIM values were more or less proportional to the highest PSNR values. The absolute highest values of objective criteria PSNR-HVS/PSNR-HVSM appeared around 1600 epochs; however, the difference between 1600 epochs and 2000 epochs was negligible. The mentioned assessment methods are described in Section 4.2.1.

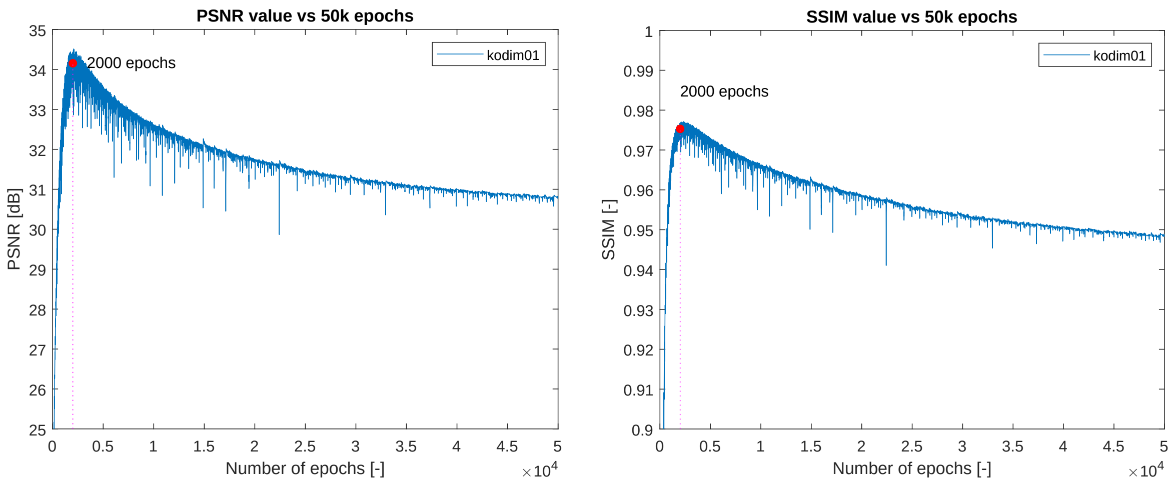


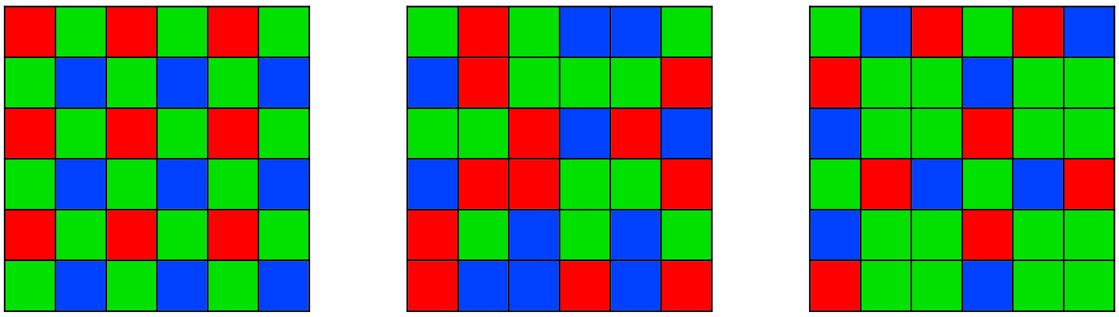
Fig. 4.1: The selection of the number of epochs. The graphs show the PSNR and SSIM values of the first image of the Kodak data set (kodim01) throughout 50000 epochs. A similar behavior was observed for other images in the data set.

4.1.2 Artificial mosaicing

Artificial mosaicing is a term to describe the process of artificially creating “RAW” images. Real “RAW” images are normally captured by the image sensor in a camera; however, for the purpose of easier testing, these images had to be simulated. This process takes an RGB image in the PNG-24 file format and converts it to an R, G, B undersampled image using an CFA mask (filter). Three types of CFA masks were employed in the experiments – the Bayer mask (Section 3.2), the X-Trans filter [35] and a random filter.

4.1.3 Usage of different filters

Most cameras use the Bayer filter. Consequently, most demosaicing methods including the ones compared with the proposed method use a demosaicing algorithm based solely on the Bayer mask. These algorithms are specifically designed for the Bayer CFA and therefore would be dysfunctional with other color filter arrays (CFAs). One of the advantages of the proposed method is that it can be used with multiple filters. As mentioned in Section 4.1.2, three filters were utilized in the experiments. The random filter is easily imaginable – randomly placed red, green and blue components. All filters can be seen in Fig. 4.2.



a) the Bayer mask b) the random filter b) the X-Trans filter

Fig. 4.2: Different color filters used in experiments.

The X-Trans filter [35] is a special CFA developed by Fujifilm and used in their X series cameras. The X-Trans CFA consists of a 6×6 pattern (see Fig. 4.2). Due to its more irregular layout, it offers a better reduction of the color artifacts (aliasing) than e.g. the Bayer mask [35].

The random filter is randomly generated before the experiments. The color distribution is 1/3 for every color value (red, green and blue).

4.2 Evaluation

As mentioned before, the proposed demosaicing algorithm was compared with other commonly used methods. The comparison was done using a variety of assessment methods. The evaluation is made possible due to the artificial mosaicing (Section 4.1.2), since the “mosaiced” image is made from an *original* fully colored image (ground truth).

4.2.1 Assessment methods

These methods include – Peak signal-to-noise ratio (PSNR), PSNR-HVS and PSNR-HVSM, and structural similarity index measure (SSIM). It is worth noting that the SSIM value is defined in an interval $[0, 1]$ with the maximum of 1. Additionally, higher PSNR values are better (the same holds for PSNR-HVS and PSNR-HVSM). All the objective scores of these methods were calculated in Matlab. SSIM and

PSNR have a built in function directly in Matlab. The Matlab code for PSNR-HVS and PSNR-HVSM was taken from [36].

Peak signal-to-noise ratio

PSNR [37] is a quality measure between an original image and a reconstructed (or compressed) image in decibels. It is the ratio between maximum (peak) power of the signal and the maximum power of the noise. The calculation uses mean-square error (MSE) and is calculated (for an 8-bit image) as

$$\mathrm{PSNR} = 10 \cdot \log_{10}\left(\frac{255^2}{\mathrm{MSE}}\right), \quad \mathrm{MSE} = \frac{\sum_{m,n}[I_1(m,n) - I_2(m,n)]^2}{M \cdot N}$$

where M and N are the number of rows and columns. I_1 and I_2 are the original and the reconstructed image.

PSNR-HVS and PSNR-HVSM

PSNR-HVS [38] computes the standard PSNR, but takes into account the Human Visual System (HVS), which is more sensitive to distortions in low frequency regions. It removes the so called mean shifting and contrast stretching using a scanning window [38]. PSNR-HVS is calculated with the same equation as the classic PSNR, only the MSE is calculated differently. The image is split into 8×8 blocks and the modified MSE is calculated using discrete cosine transform (DCT) coefficients and a matrix of correcting factors. This method is thoroughly described in [38].

PSNR-HVSM [39] is an addition to the PSNR-HVS value. It takes into account not just the HVS but also a model proposed in [39] that reduces the DCT coefficients by a contrast masking values that are calculated with the help of a contrast sensitivity function (CSF). This method is considered to be one of the best objective assessment methods for image quality. It is closely described in [39].

Structural similarity index measure

SSIM [40] is an index that models any image distortion as a combination of three factors – luminance distortion, loss of correlation and contrast distortion. The SSIM aims to replace the assessment methods that are based on the MSE. More in depth description can be found in [40].

4.2.2 Randomness of DIP

Another difference between DIP and other common demosaicing methods is that randomness of various kinds plays a role in the output of DIP-based methods. Firstly, the neural network is initialized with a random code vector z every time the optimization is run. Secondly, along with the random z, a random parameter θ (initial network weights) is generated, meaning that every optimization process possibly finds a different local minimum of the cost (loss) function. Lastly, the optimization algorithm Adam [34] leads to some randomness because of stochastic gradient descent. All of this arbitrariness amounts to slight differences in the output PSNR (PSNR-HVS/PSNR-HVSM included) and SSIM values over multiple runs of the optimization process. In contrast to this, the other mentioned demosaicing methods are purely deterministic.

4.3 Findings

This section compares the proposed method with commonly used demosaicing algortihms such as the bilinear interpolation, Malvar's method, Menon's method and the `demosaic` funtion in Matlab. It also mentions the downsides and describes any additional improvements that can be done to improve its quality.

4.3.1 Comparison in terms of different assessment methods

Multiple experiments with different CFAs were done using the proposed method. As already mentioned, all of the SSIM and PSNR (including PSNR-HVS/PSNR-HVSM) values were calculated using Matlab. Ten runs of the program on all images from the Kodak data set were averaged (due to randomness) for every CFA. The performance of the proposed method with different CFA patterns and the performance of other common demosaicing methods can be seen in Tab. 4.1.

Tab. 4.1: The average PSNR (HVS/HVSM) and SSIM values. The top part shows the performance of common demosaicing methods. The bottom part shows the performance of the proposed method with different CFAs. The best values in each part are highlighted.

	PSNR-HVSM	PSNR-HVS	PSNR	SSIM
BILINEAR	27.361	25.207	29.207	0.9245
MALVAR	34.865	31.517	35.293	0.9808
MENON	**39.238**	**35.569**	**39.076**	**0.9908**
MATLAB	32.789	30.165	34.645	0.9800
DIP – bayer	33.515	30.301	33.911	0.9678
DIP – random	35.285	31.862	34.643	0.9725
DIP – x-trans	**36.017**	**32.419**	**35.392**	**0.9750**

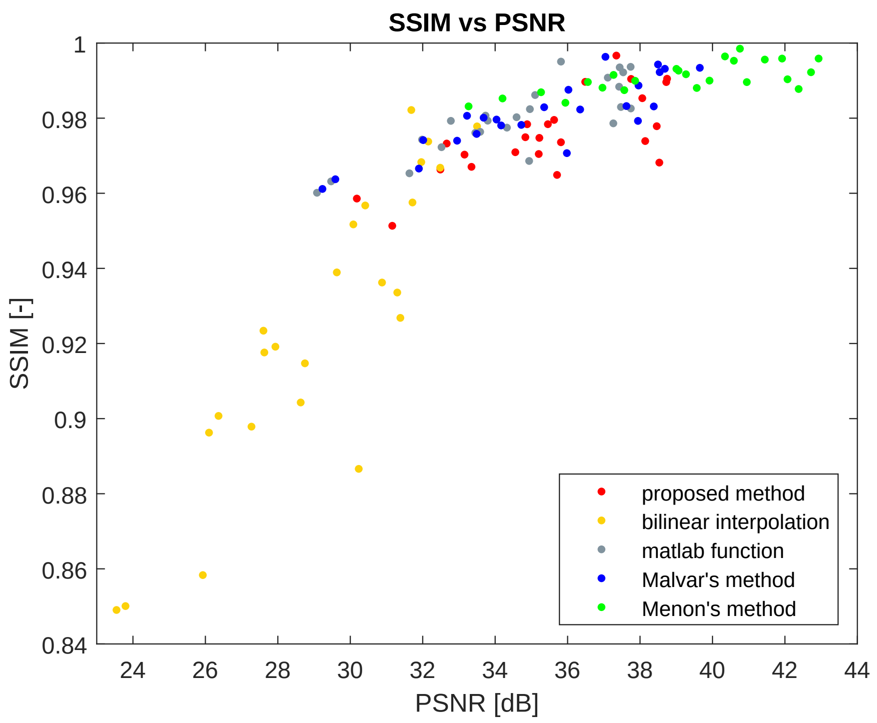


Fig. 4.3: Comparison of SSIM and PSNR values of different images from the Kodak data set. The PSNR and SSIM values for the proposed method were calculated using the X-Trans filter.

Additionally, a graphical comparison of SSIM and PSNR values of different images from the Kodak data set can be seen in Fig. 4.3.

By looking at Tab. 4.1 and Fig. 4.3 it can be seen that the proposed method performs on par with the Malvar's demosaicing method, in some cases even surpasses the Malvar's method (using the X-Trans CFA). Menon's method proved to be the best in terms of numerical comparison.

Section 4.2.2 described the effect of randomness on the output values of DIP-based methods. As for the proposed method, the difference between the best and worst run in terms of different assessment methods and different CFAs can be seen in Tab. 4.2.

Tab. 4.2: The differences between the best and worst run (out of the 10 runs) in terms of different assessment methods.

	PSNR-HVSM	PSNR-HVS	PSNR	SSIM
Bayer mask	0.927	0.782	0.570	0.0031
Random filter	**0.537**	**0.436**	**0.390**	**0.0017**
X-Trans filter	0.920	0.653	0.527	0.0018

4.3.2 Visual comparison

In the case of visual comparison, the proposed method performed significantly better then all other methods, except for Menon's demosaicing method. The proposed algorithm achieved similar visual quality to Menon's method. Additionally, images reconstructed using the X-Trans filter proved to be the same or better than the images obtained using the Menon's method. The most visible differences can be seen on the fence in the image no. 19 and the roof in the image no. 8 of the data set. The visual comparison of different demosaicing methods on the fence in image no. 19 can be seen in Fig. 4.4.

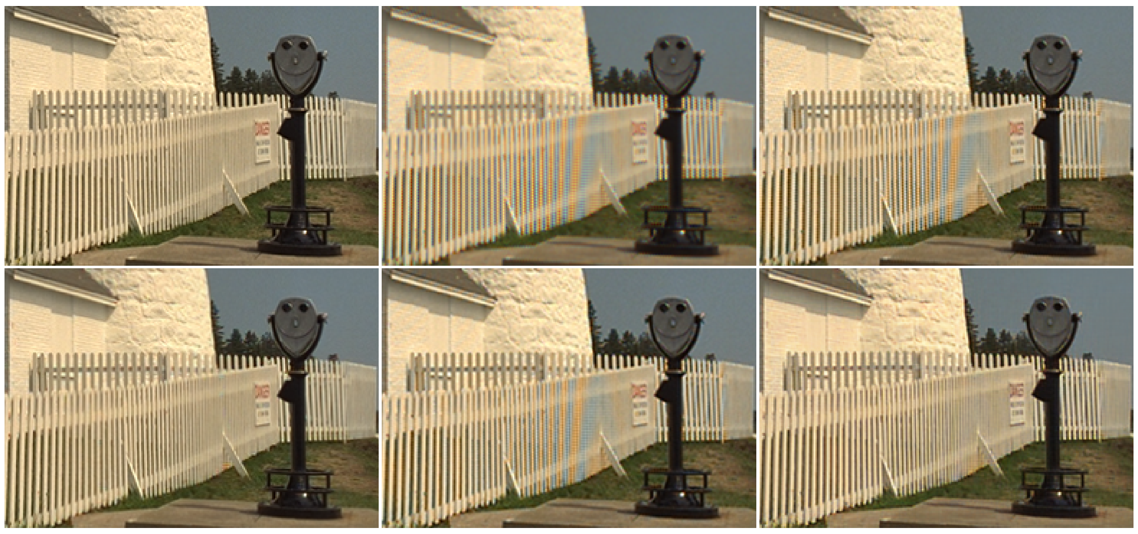


Fig. 4.4: Visual comparison of different demosaicing methods on the fence in image no. 19 of the Kodak data set. The first image in the top row represents the ground truth (original image), second image represents bilinear interpolation and the third image corresponds to the Malvar's method. The bottom row contains the Menon's method on the left, Matlab `demosaic` function in the middle and the proposed method on the right.

The visual comparison of the proposed method using different CFAs can be seen in Fig. 4.5. It is worth noting that the visual differences in the images are subjective and no formal subjective tests were performed to prove the discussed information statistically.

4.3.3 Additional improvements

The proposed method can be improved by a variety of actions. The first action is the choice of a CFA. The X-Trans filter proved to be the best out of the three filter types used in the experiments. Another and the *main* improvement that

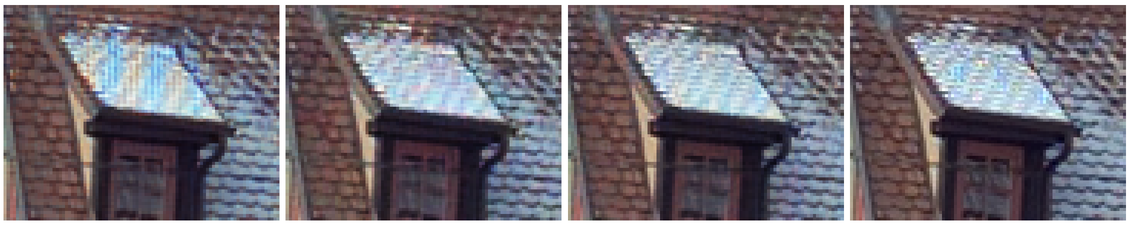


Fig. 4.5: Visual comparison between the roof in image no. 8 reconstructed using the proposed method with different filters and the Menon's method. From right to left: Bayer mask, the random filter, X-Trans filter, Menon's method.

can be done to increase the image quality is averaging the images obtained from last few epochs. Averaging just the last two images proved to increase the PSNR (HVS/HVSM) values by a significant margin. Furthermore, averaging the last 50 images proved to be the best choice with regard to the objective quality, and without any significant computational cost. The average PSNR (HVS/HVSSM) and SSIM value improvements are shown in Tab. 4.3.

Tab. 4.3: The average improvement of PSNR (HVS/HVSM) and SSIM values after averaging the last two and last 50 images.

Improvement	PSNR-HVSM	PSNR-HVS	PSNR	SSIM
Bayer mask – last 2	0.556	0.345	0.274	0.0011
Bayer mask – last 50	0.727	0.467	0.363	0.0015
Random filter – last 2	0.820	0.520	0.358	0.0013
Random filter – last 50	1.065	0.704	0.481	0.0018
X-Trans filter – last 2	0.938	0.587	0.420	0.0012
X-Trans filter – last 50	**1.220**	**0.792**	**0.554**	**0.0017**

However, averaging the images did not have a significant impact on the subjective visual quality – the image at 2000 epochs, images 1999–2000 averaged, and images 1951–2000 averaged looked identical.

4.3.4 Downsides

Along with all the positives, the proposed algorithm and DIP as a whole has one crucial downside and i.e. its computational complexity. The proposed method is (compared with other mentioned demosaicing methods) extremely demanding when it comes to the computer hardware. If the execution time is averaged across all images, the classic demosaicing methods take under one second to compute, with the fastest being bilinear interpolation at around 0.4 seconds. The computational time of the proposed method is around 90 seconds on average, i.e. immensely slower than the other methods.

The computational time can be reduced with a better graphics processing unit (GPU). The time can fluctuate depending on the power of the GPU in the testing PC. The computer used to acquire the said times had an NVIDIA Tesla V100S PCIe 32 GB graphics card. It is worth noting that the testing images were the size of 768×512, which is small compared with the size of images taken by today's cameras.

4.3.5 Surprises

As an attempt to see the maximal capabilities of the proposed method, an experiment using the oracle approach was made. This approach is made possible due to the presence of the ground truth images. The oracle approach means that the knowledge of the ground truth is utilized for the selection of the best possible image. The best image in this situation means an image with the maximal PSNR/SSIM value is picked out during the optimization process.

The biggest surprise arose with averaging the images in last few epochs. Averaging just the last two images obtained similar results to the oracle approach. To take it even further, averaging the images in last 10 epochs brought greater results than the oracle approach. The comparison between averaging images in last few epochs and the oracle approach can be seen in Tab. 4.4.

Tab. 4.4: The PSNR (HVS/HVSM) and SSIM value comparison between the oracle approach and averaging the last few images. Comparison between different CFAs. "2000 epochs" represents the PSNR/SSIM values of the image in the last epoch.

AVG	PSNR-HVSM	PSNR-HVS	PSNR	SSIM
Bayer – 2000 epochs	32.789	29.835	33.548	0.9663
Bayer – last 2 averaged	**33.345**	**30.179**	**33.822**	**0.9674**
Bayer – oracle approach	33.292	30.155	33.785	0.9672
Random – 2000 epochs	34.219	31.157	34.163	0.9707
Random – last 2 averaged	**35.039**	**31.678**	**34.520**	**0.9719**
Random – oracle approach	34.870	31.575	34.453	0.9717
X-Trans – 2000 epochs	34.797	31.627	34.838	0.9733
X-Trans – last 2 averaged	**35.735**	**32.214**	**35.258**	**0.9745**
X-Trans – oracle approach	35.489	32.064	35.151	0.9742

Conclusion

This thesis focused on the problem of demosaicing specifically demosaicing using a revolutionary concept called Deep Image Prior [3]. This concept utilized an untrained convolutional neural network (CNN) to solve different reconstruction tasks, such as denoising and inpainting. The main goal of this thesis was to implement a new demosaicing method with DIP and compare it with other commonly used demosaicing methods. The proposed method was compared with – bilinear interpolation, Malvar's method [1], Menon's method [2] and the `demosaic` function in Matlab.

The application of inpainting helped with the implementation of the proposed demosaicing method, because both problems behave similarly when considering deep image prior – both of them try to find the missing values of pixels. The key difference is in the usage of a mask that represents the missing color values. Instead of using a binary mask as in inpainting, demosaicing uses a CFA mask.

It it worth noting that deep image prior is very versatile, which means it was not expected to be the best at one specific task. However, it has shown some interesting results. In the case of numerical comparison, the proposed method was compared using a variety of assessment methods – PNSR, PSNR-HVS, PSNR-HVSM and SSIM. A visual comparison was also done; however, no formal subjective tests were performed. Additionally, the proposed method was tested with different CFAs – the Bayer mask, a random filter, and X-Trans filter.

In the numerical comparison, the proposed method performed similarly to the Malvar's demosaicing method. When using an X-Trans filter the proposed method performed even better than the Malvar's method. However, the Menon's method proved to be the superior method (considering the numerical comparison). Furthermore, averaging the last few images of the learning process brought solid results, even when compared with the oracle approach. Visually, the proposed method surpassed all other methods except the Menon's method. Its performance was on par with the Menon's method. In the case of the X-Trans filter array the proposed method performed better than every other method.

Even though the proposed method brought exceptional visual results, it would be difficult to use it in practice due to the fact that it is extremely computationally demanding when compared with other common methods.

Bibliography

[1] H. S. Malvar, L. He, and R. Cutler, "High-quality linear interpolation for demosaicing of Bayer-patterned color images," in *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*, May 2004, vol. 3, pp. iii–485.

[2] D. Menon, S. Andriani, and G. Calvagno, "Demosaicing With Directional Filtering and a posteriori Decision," *IEEE Transactions on Image Processing*, vol. 16, no. 1, pp. 132–141, Jan. 2007.

[3] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Deep Image Prior," *International Journal of Computer Vision*, vol. 128, no. 7, pp. 1867–1888, Jul. 2020.

[4] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015.

[5] S. Razavi, "Deep learning, explained: Fundamentals, explainability, and bridgeability to process-based modelling," *Environmental Modelling & Software*, vol. 144, p. 105159, Oct. 2021.

[6] L. Alzubaidi *et al.*, "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions," *Journal of Big Data*, vol. 8, no. 1, p. 53, Mar. 2021.

[7] Q. Liu and Y. Wu, "Supervised Learning," in *Encyclopedia of the Sciences of Learning*, N. M. Seel, Ed. Boston, MA: Springer US, 2012, pp. 3243–3245.

[8] "What is Supervised Learning?," *IBM*, Jun. 30, 2021. `https://www.ibm.com/cloud/learn/supervised-learning` (accessed Nov. 28, 2022).

[9] "What is Unsupervised Learning?," *IBM*, Mar. 31, 2022. `https://www.ibm.com/cloud/learn/unsupervised-learning` (accessed Nov. 28, 2022).

[10] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, "Convolutional neural networks: an overview and application in radiology," *Insights into Imaging*, vol. 9, no. 4, pp. 611–629, Aug. 2018.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems*, Dec. 2012, vol. 25, pp. 1097–1105.

[12] N. Strisciuglio, M. Lopez Antequera, and N. Petkov, "Enhanced robustness of convolutional networks with a push–pull inhibition layer," *Neural Computing and Applications*, vol. 32, pp. 1–15, Dec. 2020.

[13] E. Shelhamer, J. Long, and T. Darrell, “Fully Convolutional Networks for Semantic Segmentation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 4, pp. 640–651, Apr. 2017.

[14] O. Ronneberger, P. Fischer, and T. Brox, “U-Net: Convolutional Networks for Biomedical Image Segmentation,” in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, Oct. 2015, vol. 9351 pp. 234–241.

[15] N. Adaloglou, “Intuitive Explanation of Skip Connections in Deep Learning,” *AI Summer*, Mar. 23, 2020. `https://theaisummer.com/skip-connections/` (accessed Dec. 02, 2022).

[16] 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, Aug. 2007.

[17] A. Buades, B. Coll, and J.-M. Morel, “A non-local algorithm for image denoising,” in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)*, Jun. 2005, vol. 2, pp. 60–65.

[18] C. Ledig *et al.*, “Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017, pp. 105–114.

[19] L. Wei-Sheng, H. Jia-Bin, N. Ahuja, and M.-H. Yang, “Deep Laplacian Pyramid Networks for Fast and Accurate Super-Resolution,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017, pp. 5835–5843.

[20] M. Bevilacqua, A. Roumy, C. Guillemot, and M. A. Morel, “Low-Complexity Single-Image Super-Resolution based on Nonnegative Neighbor Embedding,” in *Proceedings of the British Machine Vision Conference 2012*, Sep. 2012, pp. 135.1–135.10.

[21] J. S. Ren, L. Xu, Q. Yan, and W. Sun, “Shepard Convolutional Neural Networks,” in *Advances in Neural Information Processing Systems*, Dec. 2015, vol. 28., pp. 901–909.

[22] V. Papyan, Y. Romano, M. Elad, and J. Sulam, “Convolutional Dictionary Learning via Local Processing,” in *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct. 2017, pp. 5306–5314.

[23] en:User:Cburnett, *English: A Bayer pattern on a sensor in isometric perspective/projection.* 2006. Accessed: Dec. 08, 2022. [Online]. Available: `https://commons.wikimedia.org/wiki/File:Bayer_pattern_on_sensor.svg`

[24] B. K. Gunturk, J. Glotzbach, Y. Altunbasak, R. W. Schafer, and R. M. Mersereau, "Demosaicking: color filter array interpolation," *IEEE signal processing magazine*, vol. 22, no. 1, pp. 44–54, Jan. 2005.

[25] T. Qiao, X. Luo, H. Yao, and R. Shi, "Classifying between computer generated and natural images: An empirical study from RAW to JPEG format," *Journal of Visual Communication and Image Representation*, vol. 85, p. 103506, May 2022.

[26] K. Murugesh and P. K. Mahesh, "Camera Raw Image Processing and Registration Using Raw CFA Images," *Turkish Journal of Computer and Mathematics Education*, vol. 12, no. 10, pp. 4376–4381, 2021.

[27] "What is a RAW file and how do you open it? | Adobe," *Adobe.* `https://www.adobe.com/creativecloud/file-types/image/raw.html` (accessed Nov. 25, 2022).

[28] R. H. Hibbard, "Apparatus and method for adaptively interpolating a full color image utilizing luminance gradients," U. S. Patent 5 382 976, Jan. 04, 1995.

[29] C. A. Laroche and M. A. Prescott, "Apparatus and method for adaptively interpolating a full color image utilizing chrominance gradients," U. S. Patent 5 373 322, Dec. 13, 1994.

[30] D. R. Cok, "Signal processing method and apparatus for producing interpolated chrominance values in a sampled color image signal," U. S. 4 642 678, Feb. 10, 1987.

[31] J. Mukherjee, R. Parthasarathi, and S. Goyal, "Markov random field processing for color demosaicing," *Pattern Recognition Letters*, vol. 22, no. 3, pp. 339–351, Mar. 2001,

[32] O. Kapah and H. Z. Hel-Or, "Demosaicking using artificial neural networks," in *Applications of Artificial Neural Networks in Image Processing V*, Apr. 2000, vol. 3962, pp. 112–120.

[33] B. K. Gunturk, Y. Altunbasak, and R. M. Mersereau, "Color plane interpolation using alternating projections," *IEEE Transactions on Image Processing*, vol. 11, no. 9, pp. 997–1013, Sep. 2002.

[34] D. P. Kingma and L. J. Ba, “Adam: A Method for Stochastic Optimization,” in *International Conference on Learning Representations (ICLR)*, 2015.

[35] M. Rafinazari and E. Dubois, “Demosaicking algorithm for the Fujifilm X-Trans color filter array,” in *2014 IEEE International Conference on Image Processing (ICIP)*, Oct. 2014, pp. 660–663.

[36] “Nikolay Ponomarenko homepage - PSNR-HVS-M download page.” `https://www.ponomarenko.info/psnrhvsm.htm` (accessed May 20, 2023).

[37] “Compute peak signal-to-noise ratio (PSNR) between images – Simulink,” *MathWorks.* `https://www.mathworks.com/help/vision/ref/psnr.html` (accessed May 16, 2023).

[38] K. Egiazarian, J. Astola, N. Ponomarenko, V. Lukin, F. Battisti, and M. Carli, “New full-reference quality metrics based on HVS,” in *Proceedings of the Second International Workshop on Video Processing and Quality Metrics VPQM-06*, Scottsdale, USA, Jan. 2006.

[39] N. Ponomarenko, F. Silvestri, K.Egiazarian, M. Carli, V. Lukin, “On Between-Coefficient Contrast Masking of DCT Basis Functions,” in *Proceedings of Third International Workshop on Video Processing and Quality Metrics for Consumer Electronics VPQM-07*, Scottsdale, USA, Jan. 2007.

[40] Z. Wang and A. C. Bovik, “A universal image quality index,” *IEEE Signal Processing Letters*, vol. 9, no. 3, pp. 81–84, Mar. 2002.

Symbols and abbreviations

1D	one-dimensional
2D	two-dimensional
3D	three-dimensional
ANN	artificial neural network
A/D	analog-to-digital
B	blue
b	bias
CapsNet	Capsule Neural Network
CFA	color filter array
CNN	convolutional neural network
CSC	convolutional sparse coding
CSF	contrast sensitivity function
D	the number of neuron inputs
d	downsampling operator
DCT	discrete cosine transform
DenseNet	Dense Convolutional Network
DIP	Deep Image Prior
E	energy
FC	fully connected
FIR	finite impulse response
f_θ	neural network function
G	green
GPU	graphics processing unit
h	height

HR	high resolution
HVS	Human Visual System
LapSRN	Laplacian Pyramid Super-Resolution Network
LR	low resolution
MAP	maximum a posteriori
MSE	mean-square error
Mpx	Megapixel
NLM	Non-local means
PDF	probability density function
PSNR	peak signal-to-noise ratio
R	red
R	regularizer
ReLU	rectified linear unit
ResNet	residual neural network
RGB	red-green-blue
SRResNet	super-resolution residual neural network
SSIM	structural similarity index measure
t	upsampling factor
TV	Total Variation
VGG	Visual Geometry Group
w	width
x	original image (ground truth)
x_0	corrupted image
x_c	image obtained by early stopping of optimization
x_e	enhanced image

x_{f}	detailed image
x^*	restored image
z	randomly initialized code vector
ΔH	horizontal gradient
ΔV	vertical gradient
$\odot$	Hadamard product
θ	parameter representing the weights and biases of neural network
θ^*	optimizer found using gradient descent