Treball de Fi de Grau

Grau en Enginyeria Física (GEF)

Grau en Ciència i Enginyeria de Dades (GCED)

Unsupervised metrics for unsupervised image denoising

Author: Adrià Marcos Morales

Director: Carlos Fernández-Granda

Tutor: Ferrán Marqués Acosta

Sitting: May 2022

Centre de Formació
Interdisciplinària Superior

NYU | Center for Data Science

Thanks to Prof. Carlos Fernández-Granda for letting me work within his team of the Center for the Data Science of the University of New York and for his direct implication in the different stages of the projects I've taken part in.

Thanks also to Matan Leibovich and Sreyas Mohan for letting me work with them on these challenging projects and for having patience to answer my questions and give me advice on the whole process of problem design and solving.

Thanks to Prof. Peter Crozier and his group from Arizona State University for his assistance and supervision as collaborators in the project.

Finally, thanks to Prof. Ferran Marqués Acosta for being my local thesis tutor and for orientation and interest on keeping track of my advances.

A més, gràcies als meus amics i família per acompanyar-me durant aquest viatge personal, per preocupar-se per mi a part de per la meva feina i per oferir-me assistència quan m'ha calgut. En especial gràcies a l'Oriol, l'Ana i a la meva família pel suport emocional i al Dani per les seves contribucions intel·lectuals a banda de la companyia i suport.

Abstract

In the recent years the extraordinary improvement of the computational resources and the Deep Learning techniques have led to a set of new methods for image denoising. These methods are mostly based on the supervised training of Convolutional Neural Networks, using clean images to generate (noisy, clean) pairs as (input, output) data during the training.

This approach restricts the application of these methods to imaging fields where clean images are available. This excludes most of the scientific imaging data, where the collection of clean images is impossible using the present techniques.

Recently, different authors have introduced a set of new techniques that do not require clean data in what has been named Unsupervised Denoising. Nevertheless, no major comparisons among these methods have been performed, so there is no clear state of the art approach.

Furthermore, the evaluation of these methods has always been performed in a supervised way, comparing the denoised results with true clean ground truth images, due to the lack of unsupervised reliable metrics.

Attending to these facts, in the present work we design and test a set of unsupervised metrics for model comparison in the absence of clean data and use them to drive a global comparison among the different methods to the date using a common dataset of Transmission Electron Microscope images.

En els darrers anys, el rapid desenvolupament dels recursos computacionals i les tècniques d'aprenentatge profund han donat lloc a un conjunt de nous mètodes per al "denoising" d'imatges. Aquests mètodes es basen principalment en l'entrenament supervisat de xarxes neuronals convolucionals, utilitzant imatges netes per generar parelles (sorollosa, neta) com a dades d'entrada i sortida durant l'entrenament.

Aquest enfocament restringeix l'aplicació d'aquests mètodes als camps d'imatge on existeixen imatges netes. Això exclou la majoria de les dades d'imatge científica, on la recollida d'imatges netes és impossible emprant les tècniques actuals.

Recentment, diferents autors han introduït un conjunt de tècniques que no requereixen dades netes en el que s'ha anomenat "Denoising" No Supervisat. No obstant això, no s'han realitzat comparacions importants entre aquests mètodes, de manera que no hi ha un mètode de referència inequívoc.

A més, l'avaluació d'aquests mètodes sempre s'ha dut a terme de manera supervisada, comparant els resultats del "denoising" amb imatges netes de referència, a causa de la manca de mètriques fiables no supervisades.

Tenint en compte aquests fets, en aquest treball dissenyem i testem un conjunt de mètriques no supervisades per a la comparació de models en absència de dades netes i les utilitzem per dur a terme una comparació global entre els diferents mètodes fins a la data utilitzant un conjunt d'imatges comú del camp de la microscòpia electrònica de transmissió.

En los últimos años, la extraordinaria mejora de los recursos computacionales y las técnicas de Deep Learning han dado lugar a un conjunto de nuevos métodos para eliminar el ruido de las imágenes. Estos métodos se basan principalmente en el entrenamiento supervisado de redes neuronales convolucionales, utilizando imágenes limpias para generar pares (ruidosa, limpia) como datos de (entrada, salida) durante el entrenamiento.

Este enfoque restringe la aplicación de estos métodos a los campos de imágenes donde hay imágenes limpias disponibles. Esto excluye la mayor parte de los datos de imágenes científicas, en los que la recopilación de imágenes limpias es imposible utilizando las técnicas actuales.

Recientemente, diferentes autores han introducido un conjunto de nuevas técnicas que no requieren datos limpios en lo que se ha denominado Unsupervised Denoising. Sin embargo no se han realizado comparaciones importantes entre estos métodos, por lo que no existe un criterio claro sobre el método de referencia.

Además, la evaluación de estos métodos siempre se ha realizado de forma supervisada, comparando los resultados sin ruido con imágenes de referencia limpias, debido a la falta de métricas fiables no supervisadas.

Atendiendo a estos hechos, en el presente trabajo diseñamos y probamos un conjunto de métricas no supervisadas para la comparación de modelos en ausencia de datos limpios y los usamos para llevar a cabo una comparación global entre los diferentes métodos hasta la fecha utilizando un conjunto de imágenes común de Microscopio Electrónico de Transmisión.

Key Words

Transmission Electron Microscopy, Image denoising, Unsupervised image Denoising, Metrics, Unsupervised metrics, Image quality, Model Comparison, Estimation, Bias, Image, Video, Deep Learning, Image processing, Neural networks, Convolutional neural networks, SSIM, Squared error, Mean squared error, MSE

Microscopia Electrònica de Transmissió, Eliminació de soroll en imatge, Eliminació no supervisada de soroll en imatge, Mètriques, Mètriques no supervisades, Qualitat d'imatge, Comparativa de models, Estimació, Biaix, Imatge, Video, Aprenentatge profund, Processament d'imatge, Xarxes neuronals, Xarxes neuronals convolucionals, SSIM, Error quadràtic, Error quadràtic mitjà

Microscopía Electrónica de Transmisión, Eliminación de ruido en imagen, Eliminación no supervisada de ruido en imagen, Métricas, Métricas no supervisadas, Calidad de imagen, Comparativa de modelos, Estimación, Sesgo, Imagen, Video, Aprendizaje profundo, Procesamiento de imagen, Redes neuronales, Redes neuronales convolucionales, SSIM, Error cuadrático, Error cuadrático medio

Contents

List of Figures

1 Introduction

Image denoising is a fundamental image processing task that does not have a definitive solution yet. It consists on the removal of the noise introduced in images during the acquisition, storage, processing or transmission processes. It becomes crucial for the posterior image analysis and processing, for the presence of noise in the data degrades the visual quality of the images and can occlude some details.

In the recent years, due to the fast upswing of the deep learning techniques on image processing, many neural networks have been trained for image denoising by providing them pairs of noisy and clean images during the training process. These datasets are often built based on a set of clean images by means of artificial noise addition, typically Gaussian.

Nevertheless, the existence of clean datasets cannot be taken for granted in all the fields. Specially in the case of scientific imaging the acquisition technique applied may be noisy by definition. An example are low exposure time microscopy images, where only a few photons, or electrons in the case of electron microscopy, are detected per each pixel, leading to an image with Poisson distributed noise.

For that reason unsupervised denoising techniques are needed, in order to provide a tool that can be applied in the absence of clean data. Not only because such data is not always available but because the artificial noise addition does not necessarily represent what a true noisy image is, for the noise models can be complex sometimes.

In the last five years a lot of different unsupervised approaches have been published in order to cover this gap and offer a way to perform image denoising when no clean data is available.

Nevertheless, all these techniques have always been trained and validated using datasets with clean images in order to check how good the results are. That has been done in order to get more reliable results, but at the same time because there was no reliable way of comparing different denoising results when no clean data was available to perform comparisons.

1.1 Objectives

The present project has several differentiated objectives, all them regarding topics around unsupervised image denoising.

First of all, we intend to collect and organize a dataset composed of Transmission Electron Microscopy images in order to use it for model training and also release it to the community after ascertaining the absence of benchmark datasets for unsupervised denoising.

We also intend to design and explode the performance of different unsupervised metrics that allow model comparison in the absence of clean data in order to assert that this can be done in the future.

Finally, we use the chosen metrics to perform a comparison among the different benchmark model architectures plus some variations of those in other to find the best performing model up to the date for Transmission Electron Microscopy data. The final objective of this block is to serve a user friendly implementation of the best model to our collaborators for them to denoise their data when needed.

1.2 Aim of the project

The first and main aim of the present work is to provide the community with reliable metrics that allow them to test future models in the absence of clean data.

Secondly, we aim to provide the community with a new benchmark dataset for unsupervised models training and testing, aiming to cover the absence of important datasets in this field that always forces the authors to use artificial noisy images generated from clean ones.

Thirdly, we aim to perform a comparison of different state of the art architectures in order to determine which is the one that works the best in our data.

Finally, we aim to provide our partner group from Arizona State University with a denoising tool that they can use on their data without any specific Deep Learning knowledge.

2 Theoretical Framework

2.1 Electron microscopy

At the beginning of the 20th century, many physicists' efforts were focused on fundamental research, which stands for the study of the fundamental particles that compose our universe. The recent discovery of the existence of electrons in 1897 by J. J. Thomson leaded to the beginning of the study of cathode rays (i.e., electron beams) and their applications.

Some years after that, French physicist Louis de Broglie suggested, in 1924, the idea that electron beams might be regarded as a form of wave motion, introducing for the first time the concept of wave particle duality, which states that electrons and other discrete bits of matter, which until then had been conceived only as material particles, show both wavelike and particle-like characteristics depending on the situation. This way, De Broglie derived the formula for the wavelength of the wave associated to a traveling electron.

This new discovering leaded to the idea of electron microscopy; if such waves could be altered by filters just like light waves are altered by lenses, then they could be used to build a electron driven microscope, which would eventually lead to a considerable increase in imaging resolution. In 1926 it was demonstrated that electromagnetic fields could serve as lenses for electrons or other charged particles. This discovery initiated the study of electron optics and, by 1931, two German electrical engineers named Max Knoll and Ernst Ruska had devised a two-lens electron microscope that produced images of the electron source. Two years later, in 1933, a first primitive electron microscope was built that imaged a specimen rather than the electron source, and in 1935 Knoll produced a scanned image of a solid surface for the first time.

Figure 1: Vladimir Zworykin, co-inventor of television, demonstrates the first electron microscope in the United States in 1940.

German physicist Manfred von Ardenne, and British electronic engineer Charles Oatley laid the foundations of the two main electron microscopy techniques, which are still used nowadays; transmission electron microscopy (TEM) and scanning electron microscopy (SEM).

TEMs employ a high voltage electron beam in order to create an image. An anode or electron gun is located at the top of the TEM and emits electrons that travel through the microscope's vac-

uum tube. Instead of having a glass lens focusing the light (as in the case of light microscopes), the TEMs employ an electromagnetic lens which focuses the electrons into a very fine beam. This beam then passes through the specimen, which is always a very thin layer of material, and the electrons either scatter or hit a fluorescent screen at the bottom of the microscope. The sample itself is mounted on a stage in the chamber area and (unless the microscope is designed to operate at low vacuums) both the column and the chamber are evacuated by a combination of pumps. This way TEMs can produce images showing the inner structure of the sample up to atomic scale, building a single-channel image where the fade of the different areas is directly related with the density of the corresponding regions of the sample.

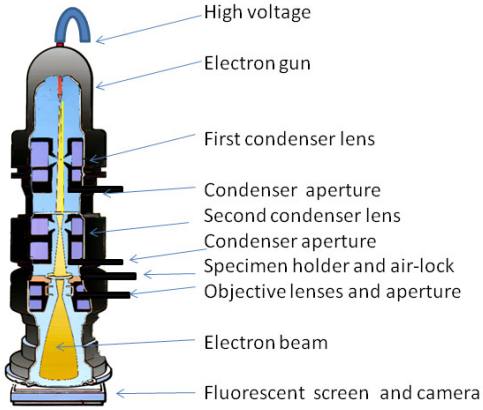

Figure 2: Simplified diagram of a transmission electron microscope.

In the case of SEMs, electrons are also produced at the top of the column, accelerated down, and passed through a combination of lenses and apertures to produce a focused beam which then strikes the surface of the sample, just like in TEMs, but here the beam has a lower energy level (the electrons carry less speed) and the beam travels around the sample. The position of the electron beam on the sample is controlled by scan coils situated above the objective lens. These coils allow the beam to be scanned over the surface of the sample, since the energy of the electrons is too low to break through it. A set of detectors collect the electrons once they have scattered after reaching the surface of the sample. This beam scanning enables information about the surface of the sample to be collected.

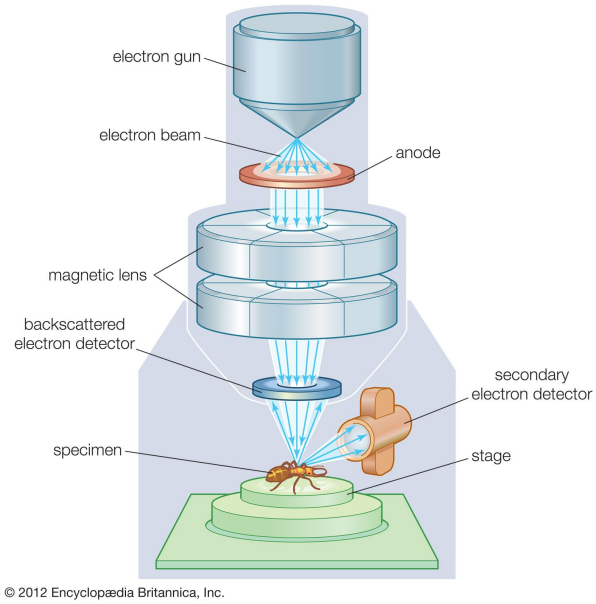

Figure 3: Simplified diagram of a scanning electron microscope.

TEMs can image specimens up to 1 micrometre in thickness and produce grayscale images corresponding to the density of the sample, and the development of modern electron optics and detectors have leaded to the modern versions of TEMs, which are capable of imaging at ultra high resolution, making atomic images possible.

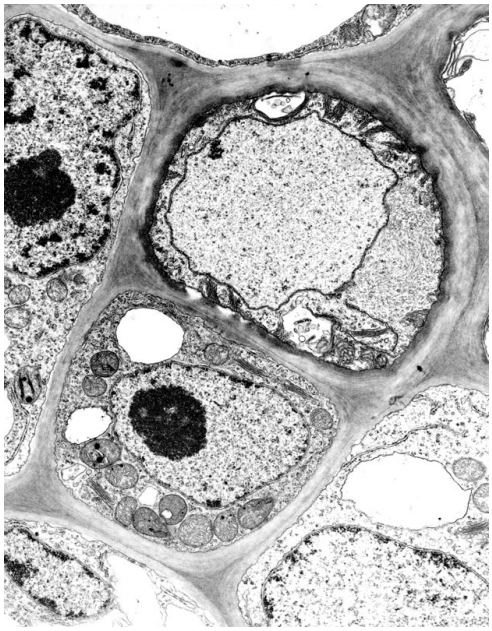

Figure 4: Cotton phloem tissue image obtained using a TEM.

SEMs, in which a beam of electrons is scanned over the surface of a solid object, is used to build up an image of the details of the surface structure up to a very small scale, despite its resolution is not as accurate as in the case of TEMs.

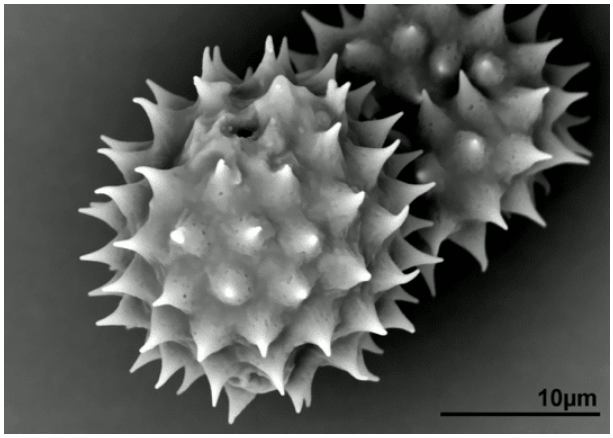

Figure 5: Pollen particle image obtained using a SEM.

2.2 Electron propagation

Different physicists, named Thompson, Reid, Davisson, and Germer, independently showed that electrons could be diffracted when passing through thin crystals of nickel and, in general, through crystals. Performing electron diffraction in TEMs was first realized by Kossel and Mollenstedt in 1939 and today it is an indispensable part of TEM and, probably, the most useful aspect for materials scientists and nanotechnologists, for whom crystal structure is an essential characteristic to control material properties.

The electron is a low-mass, negatively charged particle. As such, it can easily be deflected by passing close to other electrons or the positive nucleus of an atom. These electrostatic (Coulomb) interactions cause electron scattering, which is the process that makes TEM feasible.

A simplified way to express the idea of how electrons propagate through the atomic network is to picture each electron as a particle that goes through the atoms. Whenever its path approaches an atomic nucleus, as these are positive charged, the electron will feel a certain Coulomb attraction, curving its trajectory around it as we can see in the simplified scheme of Figure 6.

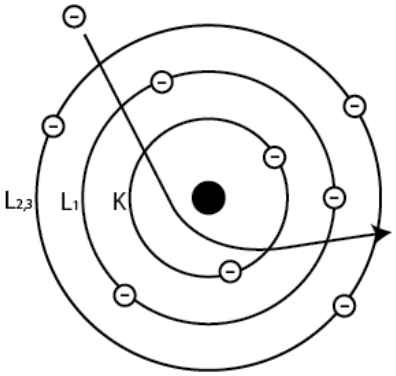

Figure 6: Scheme of an scattered electron.

In fact, there are several types of electron scattering but for the purpose of this project no further

knowledge is needed.

In the wave consideration of the electrons we talk about diffraction instead of scattering, which is the process through which an incident wave is modified when it meets an object. Diffraction is just a very special form of what is called elastic scattering.

This terminology and the particle wave duality can be confusing; Collins' Dictionary defines diffraction as 'a deviation in the direction of a wave at the edge of an obstacle in its path' while scattering is defined as 'the process in which particles, atoms, etc., are deflected as a result of collision.'. Therefore both terms are useful to describe different sources of alterations in the electron beam trajectory.

To sum up, the path followed by the electrons is constantly altered due to their interaction with the specimen, which means that once the uniform incident electron beam has gone through the sample it is no longer uniform but has more and less intense regions depending on how close from atomic nuclei the particles have travelled. In figure 7 we show a simulated image displaying the intensity of the electron beam at each region after going through a PtCe nanoparticle. This is a computer simulation built using physical models by the Crozier Research Group, from the Arizona State University.

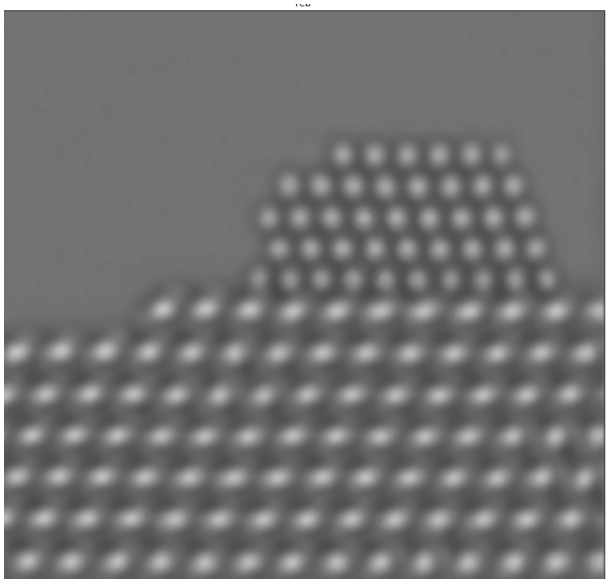

Figure 7: Simulated image of a PtCe nanoparticle

2.3 Poisson distribution

In probability theory and statistics, the Poisson distribution is a probability distribution for discrete (integer) variables. It expresses the probability of a given number of events occurring in a fixed interval of time (or space) if these events occur with a known constant mean rate and the occurrence of the event is independent of the time (or space) since the last one. It was first introduced by the French mathematician Siméon Denis Poisson, from whom receives its name, and published together with his probability theory in 1837 in his work *Recherches sur la probabilité des jugements en matière criminelle et en matière civile*.

For instance, our planet receives an average of around 500 meteorites per year, despite only a few of them reach the surface. The impacts are independent; receiving one does not change the probability of when the next one will arrive. The number of impacts received during a year has a Poisson probability distribution, since the expected number is known and the time period during which we are counting impacts is fixed.

The probability mass function of a Poisson distribution is

$$Poiss(k;\lambda) = \frac{e^{-\lambda}\lambda^k}{k!}$$

Where $\lambda > 0$ is the parameter of the distribution and is also the expected number of event occurrences in the fixed time period. The Poisson distribution is an appropriate model if the following assumptions are fulfilled:

- k is the number of times an event occurs in an interval and can take integer values, i.e. 0, 1, 2, etc.
- Events occur independently. That is, the occurrence of one event does not affect the probability that a second event will occur.
- The average rate at which events occur is independent of any occurrences.
- Two events cannot occur at exactly the same instant.

If these conditions are true, then k is a Poisson random variable, and the distribution of k is a Poisson distribution.

The last remarkable property of the Poisson distribution for this project is that the expected value and variance of a Poisson-distributed random variable are both equal to λ.

2.4 Catalysis

In chemistry, catalysis is the modification of the rate of a chemical reaction by addition of a substance that will not be consumed during the reaction. The rates of chemical reactions (the velocities at which they occur) depend upon a number of factors, including the chemical nature of the reacting species and the external conditions to which they are exposed. Catalysis is a field of interest in chemical rates, the acceleration of chemical reactions by substances not consumed in the reactions themselves (known as catalysts).

Theoretically, the study of catalysis is of interest because of what it reveals about the fundamental nature of chemical reactions. But in practice it is important because many industrial processes depend upon catalysts for their success or their efficiency.

In a catalyzed reaction, the catalyst generally enters into chemical combination with the reactants but is ultimately regenerated, so the amount of catalyst remains unchanged once the reaction has happened. Since the catalyst is not consumed it can be used as many times as wanted, so each catalyst molecule may induce the transformation of many molecules of reactants. For an active catalyst, the number of molecules transformed per minute by one molecule of catalyst may be as large as several million.

In the processes where a given reactant or a combination of them can undergo diverse reactions that yield different products simultaneously, the distribution of products may be influenced by the use of a catalyst that selectively accelerates one reaction so that it happens faster than the other. Therefore, by choosing the appropriate catalyst a particular reaction can be made to occur to the extent of practically excluding another. Many important applications of catalysis are based on selectivity of this kind.

A catalysts can be homogeneous if it coexists in a mixture with the reactants like, for example, different liquids, or heterogeneous where the catalyst is clearly distinguishable from the reactants. In this later group is where substrate catalysts are classified.

Substrate catalysts are solid materials, such as atomic networks, that perform as catalysts, generally for gas or liquid reactors reactions. These materials possess certain surface peculiarities that ease the reaction, for example trapping certain kind of reactor atoms such that they become static and the other reactors find it faster, or energetically favouring certain molecular configurations to the molecules stacked on their surface.

2.5 Deep Neural Networks

Neural Networks (NN) are the basis of the Deep Learning models, a non-linear model that, based on a huge number of trainable parameters, allows us to emulate almost any type of function. These models can be used to solve both regression problems, where the inputs are mapped to a continuous output, and classification problems where the inputs are mapped to a score for each one of a set of predefined classes.

The internal functioning of NN is in essence as follows: an input is received and propagated through the layers of the network, these layers are composed of neurons, the basic units of neural networks. Each of these neurons perform a very simple calculation with the value that has received; usually multiplying it by a weight and adding a bias, both of them parammeters of the NN, or making it go through a predefined activation function. This activation function is responsible of applying non-linearity to the model and several options are present in the literature, being the most widely used one $ReLu(x) = max(0, x)$.

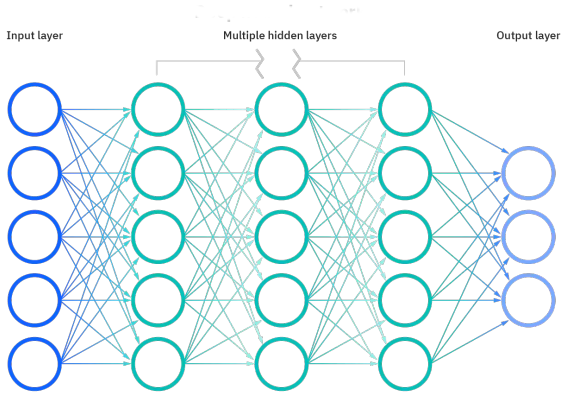

Figure 8: Simple scheme of a small Neural Network

As for the weights, they are the ones that will be modified during training, in order to bring the

output of the model when it receives a given input closer to the desired, or target, output. Deep Neural Networks (DNN) are nothing more than a Neural Network with several hidden layers, which is the name that we give the layers that are neither the input nor the output ones.

The NN architecture design is an important part of the Deep Learning approach. Although a single-layer NN can have as many neurons and, therefore, as many parameters as we want, the advantage of using deeper NNs is that we can model more complex functions with fewer weights. The ratio behind that shows us how, with more than one layer, the neurons needed to represent any type of function are reduced exponentially.

The main advantage of using DNNs is that the extraction of features for the model to perform well is done automatically from a set of input and expected output pairs. Therefore, the amount of prior work to be done over the data to extract the good criteria is drastically reduced and, in some cases, even deleted.

Generally, using Deep Neural Networks of sufficient depth, no exact prior design is needed because the network itself can extract the features that it needs to perform well. Nevertheless some smart decisions on the design of the network or its training process can, in some non-typical problems, boost the performance importantly.

2.5.1 DNN training through backpropagation

When first hearing about this kind of models, the training process through which their parameters are extracted may seem a very complex witchcraft task, but it's not. The commonly used algorithm is the backpropagation of the loss function (Rumelhart, Hinton, & Williams, 1986) consisting on the design of a loss function, also called cost or error function, which measures how bad the model is performing and, during the training process, updating the parameters in the direction of the gradient of this loss function by means of Gradient Descent technique in order to decrease that value.

Let $L(x)$ be the loss function (or error function) applied over the model output and let $W^{(n)}$ be the set of parameters of the model at the stage (n) of the training process. Let $\lambda \in \mathbb{R}^+$ be a Learning Rate. At each training iteration (or epoch), the weights are updated in the following way

$$w_{i,j}^{(n)} = w_{i,j}^{(n-1)} - \lambda \frac{\partial L(x)}{\partial w_{i,j}}$$

This is a very simple yet computationally complex process due to the presence of a huge number of partial derivatives computation, but this a priory prohibitive number of complex partial derivatives computations can be solved by the Backpropagation technique.

The backpropagation algorithm consist of two parts:

- Forward Step: Calculation of the network output, which is used to compute the value of the loss function. As the input is propagated through the different layers of neurons we get intermediate results. Let y_i be the output value of the i-th layer of the network and let $h(\cdot)$ be the activation function applied afterwards. The value of y_i can be computed as:

$$z_i = w_i^T \cdot y_{i-1}$$

$$y_i = h(z_i)$$

- Backward Step: loss backpropagation and parameter update, that is, the computation of the derivative of the loss function with respect to each one of the weights. This computation can be performed algorithmically in a recursive way. Let us first do it for the last layer weights $\{w_l, j\}$:

$$y_l = h \circ w_l^T \cdot y_{l-1} = h \circ \sum_j w_{l,j} \cdot y_{l-1,j}$$

$$L(y_l) = L \circ h \circ \sum_j w_{l,j} \cdot y_{l-1,j}$$

Using the chain rule we get the following:

$$\frac{\partial L(y_l)}{\partial w_{l,j}} = \frac{dL(y_l)}{dy_l} \cdot \frac{\partial y_l}{\partial w_{l,j}} = \frac{dL(y_l)}{dy_l} \cdot \frac{dh}{dz_l} \cdot \frac{\partial z_l}{\partial w_{l,j}} = \frac{dL(y_l)}{dy_l} \cdot h'(z_l) \cdot y_{l-1,j}$$

Let us define $\delta_l = \frac{dL(y_l)}{dy_l} \cdot h'(z_l)$ so that we can simplify to:

$$\frac{\partial L(y_l)}{\partial w_{l,j}} = \delta_l \cdot y_{l-1,j}$$

In a similar way we could extract that

$$\frac{\partial L(y_l)}{\partial w_{l-1,j}} = \frac{dL(y_l)}{dy_l} \cdot \frac{dy_l}{dz_l} \cdot \frac{\partial z_l}{\partial y_{l-1}} \cdot \frac{dy_{l-1}}{dz_{l-1}} \cdot \frac{\partial z_{l-1}}{\partial w_{l-1,j}} = \delta_l \cdot w_l^T \cdot h'(z_{l-1}) \cdot y_{l-2,j}$$

Hence we see a recursively propagating backwards; we can see how each one of the factors used to compute the derivative at the last layer is used for the previous one, and in the same way each layer will use terms from all those it has ahead.

This way we can simply implement some derivatives for the different functions inside our network and the gradient with respect to each of the neurons will be extracted from this backpropagation process.

2.5.2 Convolutional Neural Networks

The agenda for Artificial Intelligence in Imaging is to enable machines to view the world as humans do, detect and recognize features and use these abilities for tasks such as Image & Video recognition, Image Analysis & Classification, Media Recreation, Recommendation Systems, Natural Language Processing, etc. Almost all the advances achieved in Image processing due to Neural Networks are based in the architecture of Convolutional Neural Networks.

A Convolutional Neural Network (CNN) is a Deep Learning algorithm which can take an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a CNNs is much lower as compared to other more classical classification algorithms since these can learn the important filters to recognize target characteristics automatically.

The architecture of a CNN is analogous to that of the connectivity pattern of Neurons in the Human Brain region known as the Receptive Field, in the Visual Cortex. Here a collection of such fields overlap to cover the entire visual area, each one being able to detect more complex features than the previous one.

CNNs are composed of a succession of filters or kernels that are applied through convolutions on the image. This way the detection that these filters do is transition invariant, which is equivalent to say that the network is able to detect a given feature no matter where it is placed inside the image.

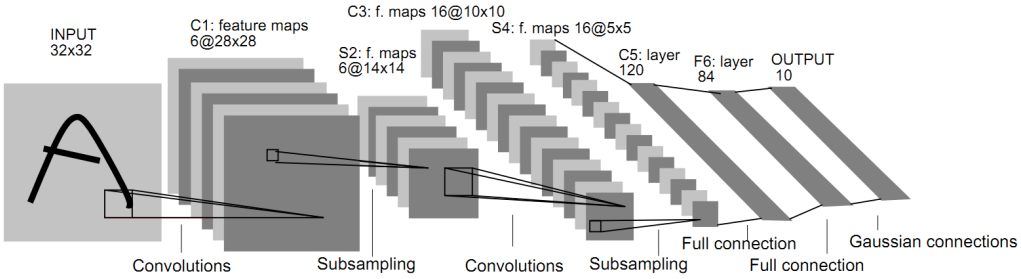

Figure 9: Simple scheme of a Convolutional Neural Network

Traditionally the convolutional layers in CNNs are paired with nonlinear activation functions, being ReLu $(max(0, x))$ the most used one, and alternated with pooling layers, where the dimensions of the image are reduced by a given factor by combining the values of neighbouring pixels. These two kinds of layers may seem very simple but obey to very important purposes: Activation functions allow the network to give importance to the output of certain filters and ignore some others, while pooling layers allow, by reducing the scale, to focus in wider regions without building bigger convolutional kernels.

This way the fist layers of kernels learn to detect very simple patterns such as lines and dots and the later ones use these first to compose more complex shapes, just as the neurons in the Visual Cortex do.

In later CNNs implementations, designed such that the output is also an image, upsampling strategies have been designed and are applied after a set of padding or down-sampling steps. The most popular strategy for this is known as inverse convolution and consists in multiplying a kernel matrix by each pixel value to extend it to different pixels.

2.5.3 Network's Jacobian

Let $x \in \mathbb{R}^M$ be an image and let $F : \mathbb{R}^M \to \mathbb{R}^N$ be a CNN that maps x to an output $F(x)$.

We call the Jacobian of the CNN $F(x)$ to the partial derivative of a each item of $F(x)$ with respect to x. That way $J_{i,j} = \frac{\partial F_i}{\partial x_j}$.

Generally the whole Jacobian is never used, for it consists of a huge number of elements since M and N use to be large. Instead only a row of the Jacobian matrix is used $J_{i,.}$. That way the

partial derivatives of a given output pixel are computed with respect to each input pixel.

The Jacobian of a CNN is one of the most common tools used to discover what is the CNN paying attention to when computing a given output pixel.

2.6 Image denoising

In the field of digital images generation, compression and transmission the presence of defects in the image acquisition technique, the object captured and its environment, the transmission channel or other factors can lead to a contamination of the images by noise, distorting the image contents and losing some of the image information. The presence of noise can affect directly to the performance on subsequent image processing tasks, such as object detection, video processing, image analysis or motion tracking. Therefore, the suppression of that noise, or image denoising, plays an important role in modern image processing systems.

Image denoising is then a challenge consisting in the removal of the noise from a noisy image in order to restore the true underlying clean image. However, as edges and textures are high frequency components, same as noise, it is a difficult task to design procedures that automatically distinguish them, so the denoising process uses to involve the loss of some details.

In fact, from a mathematical perspective, image denoising is an inverse problem, implying that its solution is not unique. That is why there exist lots of different techniques to remove the noise using different approaches in the literature. For the purpose of this project we only need to understand the *Spatial Domain Filtering Techniques* (*SDF*)

2.7 Spatial Domain Filtering Techniques

SDF methods address the problem of noise removal by calculating the value of each pixel as a combination of the pixel values present in the original image.

The first filters ever used were the called linear filters, which compute each pixel as a linear combination of the original values, typically by means of convolution. The main defect of these filters is that they fail to preserve image textures. A typical example of linear filtering is the mean filter (Gonzalez, Woods, & Masters, 2009) which is typically applied in the presence of Gaussian noise and computes each pixel as the mean of the patch surrounding it in the original image.

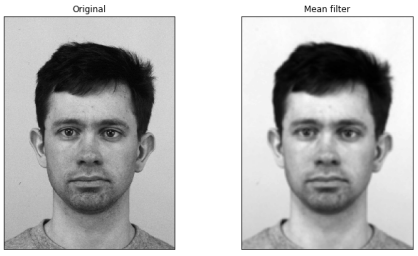

Figure 10: Example of mean filtering on Gaussian noise.

Other linear filters are Wiener filters (Benesty, Chen, & Huang, 2010) and Gaussian filters (Getreuer, 2013), but they also tend to blur sharp edges (Bergholm, 1987).

The alternative is to use non-linear or adaptive filters such as median filtering (Pitas & Venetsanopoulos, 2013) and edge detection based filters (Deng & Cahill, 1993) which use information on the morphology of the image to adapt the values taken into account to respect details or edges. A well known edge-preserving and non-linear filter for noise reduction is Bilateral filtering (Fleishman, Drori, & Cohen-Or, 2003) which performs mean filtering in a region stopping at the detected edges.

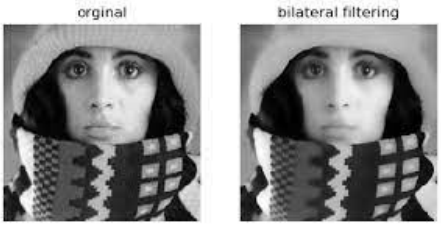

Figure 11: Example of bilateral filtering on Gaussian noise

2.8 Peak Signal-to-Noise Ratio

Peak signal-to-noise ratio (PSNR) is a metric that measures the ratio between the maximum possible power (mean squared value) of a clean signal and the power of some noise present that disturbes the signal. Because many signals have a very wide dynamic range, that is the span between the minimum and maximum possible values of the signal, the PSNR is usually expressed in terms of the logarithmic decibel scale. It is then computed as

$$\text{PSNR} = 20 \cdot \log_{10} \left(\frac{\max(s)}{\text{MSE}} \right)$$

Where $\max(s)$ is the maximum value that the signal gets and MSE is the mean squared error between the noisy and clean signals.

The larger value the PSNR gets, the less important the noise is when compared with the underlying clean signal. Using a given set of tests images, different image enhancement algorithms can be compared using PSNR to identify whether a particular algorithm produces better results.

2.9 Structural Similarity

The Structural Similarity (SSIM) is a perceptual metric originally designed to quantify the image quality degradation caused by processing the data by, for example, compression or transmission through a noisy channel. When considered in the opposite direction, SSIM can be used to determine how close a reconstructed image, by means of denoising for example, is from the real signal it is trying to represent.

SSIM requires two images from the same image capture; a reference clean image and a damaged or reconstructed one. Unlike PSNR, SSIM is based on visible structures in the image. It is therefore a good measure of perceptual image quality.

It considers image degradation as perceived change in structural information, while also incorporating important perceptual phenomena, including both luminance masking and contrast

masking terms.

The SSIM formula comes directly from multiplying the luminance, contrast and structure values of the image:

$$luminance(x,y) = \frac{2\mu_x\mu_y + c_1}{\mu_x^2 + \mu_y^2 + c_1}$$

$$contrast(x,y) = \frac{2\sigma_x\sigma_y + c_2}{\sigma_x^2 + \sigma_y^2 + c_2}$$

$$similarity(x,y) = \frac{\sigma_{xy} + c_3}{\sigma_x\sigma_y + c_3}$$

Then, taking $c_3 = c_2/2$:

$$\text{SSIM}(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}$$

Where μ_i, σ_i are the average and variance of $i = x, y$, σ_{xy} the covariance of x and y, L the dynamic range of the pixel-values and $k_1 = 0.01$ and $k_2 = 0.03$ by definition. In addition, $c_1 = (k_1 L)^2$ and $c_2 = (k_2 L)^2$ are two real variables to stabilize the division when the denominator is too small.

2.10 Spearman's rank correlation coefficient

In statistics, Spearman's rank correlation coefficient, often denoted by the Greek letter ρ, is a nonparametric measure of rank correlation that measures the statistical dependence between the rankings of two variables. In other words, it assesses how well the relationship between two variables can be described using a monotonic function.

The Spearman correlation between two variables is computed as the Pearson correlation between the rank values of those two variables; that is, given a set of value pairs, compute the correlation between the rank that each pair gets when we order them with respect to the first value and the one that it gets when we order them according to the second. While Pearson's correlation assesses linear relationships, Spearman's correlation assesses monotonic relationships, no matter if they are linear or not. In the absence of repeated data values, a perfect Spearman correlation of 1 (or -1) occurs when the variables are a perfect increasing (or decreasing) monotone function of each other.

It can be obtained as follows: given two variables X and Y stored together, we will call $R(X)$ the set of ranks we get from ordering the data with respect to X and $R(Y)$ the same when we order with respect to Y. Then, the Spearman's rank correlation coefficient can be computed as:

$$r_s = \rho_{\mathrm{R}(X),\mathrm{R}(Y)} = \frac{\mathrm{cov}(\mathrm{R}(X),\mathrm{R}(Y))}{\sigma_{\mathrm{R}(X)}\sigma_{\mathrm{R}(Y)}}$$

3 Related work and current State-of-the-Art approaches

3.1 Deep Learning filtering: CNN

In the last decades the fast increase in computational resources and power has leaded to a new whole set of denoising techniques based on Deep Learning models. CNNs are the perfect candidate to be used as denoising models (Zuo, Zhang, & Zhang, 2018) due to their structure based on convolutions, the same procedure used in classical image denoising methods.

In CNNs the supperposition of several convolutional filters could a priory be seen simply as the usage of a bigger linear filter. Nevertheless, the presence of bias addition and the activation functions turn these models into nonlinear filters, as each pixel is computed as a combination of the filters that go through the activation functions. Therefore we can consider CNNs as the most modern example of adaptive filtering due to the fact that the region used to compute each pixel's value changes depending on the content and surroundings of this pixel.

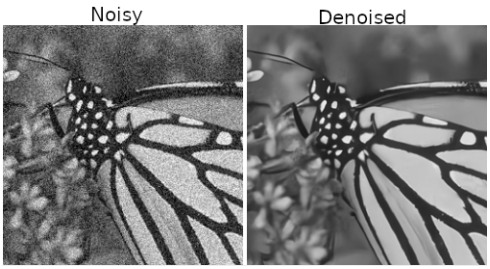

Figure 12: Example of CNN performance on Gaussian noise

Nevertheless CNNs and Neural Networks in general have the problem of requiring a large set of input-output pairs to be used during the training. This means that to train a denoising CNN we need pairs of matching clean and noisy images, which is not always available.

In TEM images, for example, clean data is virtually impossible to collect since the exposure times required to get a clean image need to be very large and the atomic systems' stable states tend to have a relatively short lifetimes. Therefore in some fields the only way to train denoising models is by using synthetic images, and these are not always easy to generate.

3.2 Unsupervised denoising using CNNs

In the past years some state of the art methods have been developed in order to train denoising CNNs in the absence of clean data based on different assumptions on the noise behaviour and the underlying signal properties. Most of them are based in the following idea:

Let's consider a noisy image $x \in \mathbb{R}^M$ and define its expected value c such that $x = c + n$ where n is the noise, which is random, independent and zero-centered ($\mathbb{E}[n] = 0$). Notice that this does not imply that the noise is additive, since any random, independent noise that fulfills $\mathbb{E}[x] = c$ will also fulfill the condition. We can then easily prove that:

$$\mathbb{E}||f(x) - x||^2 = \mathbb{E}||f(x) - c||^2 + \mathbb{E}||n||^2$$

Let's now consider $f : \mathbb{R}^M \to \mathbb{R}^M$ to be a function with trainable parameters such that $f(x)_i$ (the i-th pixel of $f(x)$ is independent of x_i (the i-th pixel of x). As this condition implies that the identity is not learnable, and taking into account that the noise is not predictable, if f is trained with the MSE loss the equality above will imply that $f(x)$ will tend to c.

In other words, since the noise is independent, random and zero centered, if a model is trained to predict the pixel values without using them it will never be able to recover the noise n and will therefore only predict the expected value c.

3.2.1 Noise2Noise

In 2018 (Lehtinen et al., 2018) was the first self supervised method for image denoising that achieved state-of-the-art performance without needing a training set with clean images. In this paper the method proposed was to train a standard CNN on pairs of noisy realizations of the same underlying image.

The idea behind this approach is the one introduced at the beginning of this section (3.2). Here they achieve independence between $f(x)_i$ and x_i by using these pairs of different realizations of a same image as input and target during the training of a CNN. That way the loss is computed as $||f(x_1) - x_2||^2$

Nevertheless this network has a very relevant problem, which is the fact that the acquisition of noisy pairs of the same image is not always possible, more in scientific imaging where the generation of images can be expensive and the reality captured on them can change quick enough for it to be impossible to capture two realizations.

3.2.2 Noise2Void

In the same year Noise2Noise was released a new algorithm appeared trying to take the idea one step further. Noise2Void (Krull, Buchholz, & Jug, 2018) introduces for the first time what they call *BlindSpot denoising*, a technique based on the same principle where the assumptions for the noise being zero-centered and independent, now among pixels instead of realizations, lead to the aim to predict the noisy value of the pixels and expect this to lead to a denoiser model.

Here, instead of training on pairs of images trying to predict one from the other, what we do is to mask some pixels using random substitutions with replacement and train the network to predict the true values of those pixels. This new idea solves the problem of needing pairs of images that Noise2Noise had but has a new problem, since the random substitution introduces artifacts to the original image at the same time that has a non-null probability of substituting a given value by itself.

3.2.3 Noise2Self

In 2019 the BlindSpot Denoising method was updated in Noise2Self (Batson & Royer, 2019). This method assumes again that the clean value of each pixel is also the expected value of the noisy one, that is that the noise is zero-centered, and that the noise in the different noisy pixels is completely independent while the underlying signal has some correlation to build a strategy to train the denoiser on the same noisy image.

Under these assumptions the network can be trained to predict the value of each noisy pixel having it masked at the input, so that the trivial solution cannot be reached. This can be achieved by

forcing the convolution to ignore the central pixel instead of modifying the image like Noise2Void does. If MSE loss, or any other metric that reaches its minimum at the expected value, is used then the network is expected to learn to predict the expected value of the pixels, which should be the clean one.

This method outperforms Noise2Void and has a more efficient training, since all the pixels are used in the BlindSpot process. Its only problem is the need of a large enough set of images with similar characteristics, but that is rarely a problem.

3.2.4 Self2Self

The last relevant method introduced in this progression is Self2Self (Quan, Chen, Pang, & Ji, 2020), published in 2020, which purposes to randomly split the noisy image pixels in two sets in order to have two masked versions with completely independent pixels and train a network to predict each one from the other. By repeating this process enough times a whole dataset can be generated from a single image. Once the network is trained, another set of masked versions can be fed in it obtaining a set of outputs and an estimation of the clean image can be obtained by averaging all is set.

This technique is very poorly efficient but extends the previous ones to the case where there is not any available set of images for train but a single one. With this last incorporation all the cases are covered.

3.2.5 UDVD

UDVD (Mohan et al., 2021) is a video denoising network based on the BlindSpot architecture using a BlindSpot-UNet network. It is trained to predict the values of the noisy pixels not only using the values of the pixels surrounding the one to be predicted in the same frame, but also those in temporal adjacent frames. It therefore extends the BlindSpot idea introduced in Noise2Self (Batson & Royer, 2019) to the temporal neighbourhood apart of the spatial used in that previous paper.

The performance of UDVD is comparable to the supervised state-of-the-art networks existing in the literature, even when trained only on a single short noisy video. This makes sense since the BlindSpot strategy, needing nothing else than the noisy data, can be fine tuned or even trained from scratch using the data that has to be denoised at each moment. This way the output of the network is the nearest possible to the expected value, namely the clean one.

3.2.6 Neighbor2Neighbor

Neighbor2Neighbor, published in 2021, is the last approach of the progression started with Noise2Noise and intends to address the case where a single image is available.

In the paper the authors purpose generating an extensive set of image pairs by down-sampling the original image by a factor of two. Given that in a 2×2 square of pixels there exist 12 different pairs of pixels to take, for an image of size $N \times N$ a total of $12^{\frac{N \times N}{4}}$ different image pairs can be generated. This can be seen in Figure 13, extracted from (Huang, Li, Jia, Lu, & Liu, 2021).

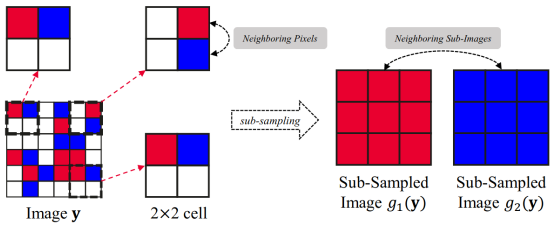

Figure 13: Scheme of the image pair generation proposed in Neighbor2Neighbor.

By doing this, and assuming that the smallest details of the images are at least two pixels long, two subsamplings of an image can be considered as a pair of realizations of the same underlying process, which is enough to apply Noise2Noise.

Despite this technique being proposed to address the denoising problems where a single noisy image is available, we extend it to large datasets by applying the same idea to all the images that compose them.

3.2.7 GainTuning

In 2021 a new method for single image denoising in the absence of a training dataset was introduced in (Mohan et al., 2021). The main idea is to take a pretrained denoising network that may be sub-optimal in the current data and adapt it to the image to be denoised at every moment. To do so the authors purpose to train only the weights used in the batch normalization in order to keep the features that the network detects but weight them differently. To train the network, a loss function based on the Stein's Unbiased Risk Estimator (SURE) is used (Metzler, Mousavi, Heckel, & Baraniuk, 2020).

The SURE is extremely useful as it estimates in an unbiased way the error of the denoiser without needing the Ground Truth Image, only using the noisy and denoised ones, in the case of Gaussian noise.

Despite being highly inefficient, the results of this method are extremely good and SURE itself is a very good tool when the trying to estimate the error in the absence of clean data, providing us with a way to evaluate a given model and compare it with others after running them in real images.

3.3 Stein's unbiased risk estimate

In 1981 Charles Stein introduced what has been popularized as Stein's unbiased risk estimate (SURE) (Stein, 1981), a simple lemma about the standard normal distribution. Indeed, Stein knew of this result much earlier and wrote about it in previous papers, but in (Stein, 1981), the author developed a multivariate extension of this lemma that led to a remarkable result on unbiased risk estimation.

This estimator provides us with a powerful tool to estimate the Mean Squared Error (MSE) of an estimate of a signal without the knowledge of that signal itself. Here we provide the formal statement.

Let $x \in \mathbb{R}^n$ be a noisy signal whose expected (and clean) value is μ and let the noise be independent and normally distributed with variance σ^2. Let $d(x)$ be an estimator of the true value of x, μ, and define $g(x)$ s.t. $h(x) = x + g(x)$. Let $g(x)$ be weakly differentiable.

Then SURE is given by:

$$SURE(h) = -n\sigma^2 + \|g(x)\|^2 + 2\sigma^2 \sum_{i=1}^{n} \frac{\partial h_i(x)}{\partial x_i}$$

proof. We wish to show that

$$\mathrm{E}_\mu \|h(x) - \mu\|^2 = \mathrm{E}_\mu\{\mathrm{SURE}(h)\}$$

We start by expanding the MSE as

$$\begin{aligned}
\mathrm{E}_\mu \|h(x) - \mu\|^2 &= \mathrm{E}_\mu \|g(x) + x - \mu\|^2 \\
&= \mathrm{E}_\mu\{\|g(x)\|^2 + \|x - \mu\|^2 + 2g(x)^T(x - \mu)\} \\
&= \mathrm{E}_\mu \|g(x)\|^2 + \mathrm{E}_\mu \|x - \mu\|^2 + 2\,\mathrm{E}_\mu\{g(x)^T(x - \mu)\} \\
&= \mathrm{E}_\mu \|g(x)\|^2 + n\sigma^2 + 2\,\mathrm{E}_\mu\, g(x)^T(x - \mu).
\end{aligned}$$

Now we use integration by parts to rewrite the last term. We will use the fact that, by definition, $g(\mu) = 0$

$$\begin{aligned}
\mathrm{E}_\mu\, g(x)^T(x - \mu) &= \int_{x \in \mathbb{R}^n} p(x|_\mu) \times g(x)^T(x - \mu) \\
&= \int_{x \in \mathbb{R}^n} \frac{1}{\sqrt{2\pi\sigma^{2n}}} \exp\left(-\frac{\|x - \mu\|^2}{2\sigma^2}\right) \sum_{i=1}^{n} g_i(x)(x_i - \mu_i) d^n x \\
&= \int_{x \in \mathbb{R}^n} \frac{1}{\sqrt{2\pi\sigma^{2n}}} \exp\left(-\frac{\|x - \mu\|^2}{2\sigma^2}\right) \sum_{i=1}^{n} \frac{g_i(x)}{(x_i - \mu_i)}(x_i - \mu_i)^2 d^n x \\
&= \int_{x \in \mathbb{R}^n} \frac{1}{\sqrt{2\pi\sigma^{2n}}} \exp\left(-\frac{\|x - \mu\|^2}{2\sigma^2}\right) \sum_{i=1}^{n} \frac{g_i(x) - g_i(\mu)}{(x_i - \mu_i)}(x_i - \mu_i)^2 d^n x \\
&= \sigma^2 \sum_{i=1}^{n} \int_{x \in \mathbb{R}^n} \frac{1}{\sqrt{2\pi\sigma^{2n}}} \exp\left(-\frac{\|x - \mu\|^2}{2\sigma^2}\right) \frac{dg_i}{dx_i} d^n x \\
&= \sigma^2 \sum_{i=1}^{n} \mathrm{E}_\mu \frac{dg_i}{dx_i}
\end{aligned}$$

Substituting this into the expression for the MSE, we get that

$$\mathrm{E}_\mu \left\| h(x) - \mu \right\|^2 = \mathrm{E}_\mu \left(d\sigma^2 + \left\| g(x) \right\|^2 + 2\sigma^2 \sum_{i=1}^{d} \frac{dg_i}{dx_i} \right) = \mathrm{E}_\mu \{ \mathrm{SURE}(h) \}$$

Therefore SURE is a good tool to estimate the error of a prediction under the assumption that the noise is Gaussian and has a known and fix standard deviation. These are two very restrictive assumptions since, even though the standard deviation can be easily estimated under the Gaussian assumption, this last one is harder to fulfill.

Nevertheless this estimator can be applied outside the Gaussian noise field and still provides a good enough estimation of the risk, despite in that scenario no solid proof stands for its unbiasedness.

4 Problem description

4.1 TEM images for catalytic processes

In this project we work on image denoising for transmission electron microscope (TEM) images in collaboration with the Crozier Research Group, from the Electron Microscopy for Energy and the Environment of the Arizona State University.

Professor Peter A. Crozier has extensive experience in developing and applying advanced transmission electron microscopy techniques to problems related to energy and the environment with special emphasis on electroceramics, catalytic materials and atmospheric aerosols. He has 20 years of experience in developing and applying the technique of advanced transmission electron microscopy to problems in catalytic materials and oxide electrolytes.

This project is embedded in the framework of the study of catalytic materials, our data belonging mainly to the process of reduction of carbon monoxide (CO) using PtCe nanoparticles as substrate catalysts. The current approach to this study that the Crozier Research Group is taking involves the use of transmission electron microscopes at high density beams and low imaging period in order to capture fast videos (up to 75 f.p.s.) of the nanoparticle during the reduction reaction.

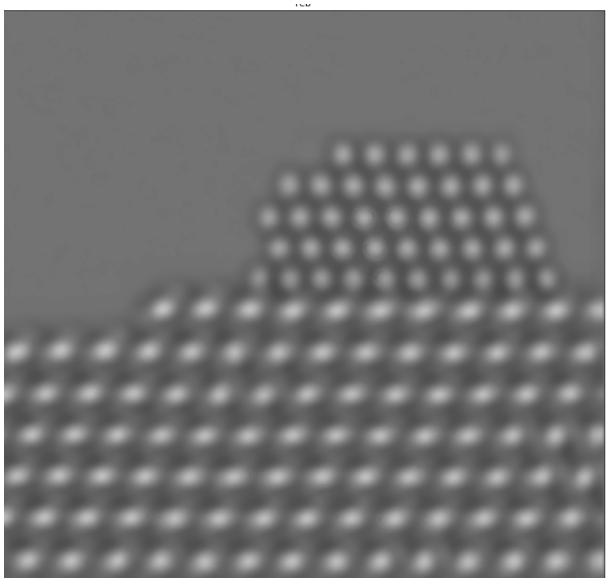

Figure 14: Simulated image of a PtCe nanoparticle

This way, the goal is to get images of the actual reaction taking place on the surface of the nanoparticle during the catalysis process. To the moment no CO particles have been observed in the images studied by this or any other groups, probably due to their speed relative to the image period or to the noise present in the images.

The visualization of these molecules is crucial to understand the physical events occurring during the reaction, both to observe and study them and to validate the theoretical models. In the same way, it is necessary to correctly visualize the structure of the nanoparticle and the substrate where this is placed in order to study the transitions between atomic configurations within them

(known as macrostates) as the energetic differences between these macrostates may establish a relationship between their transitions and the energy exchanged by a reacting molecule at the surface of the nanoparticle.

The most noticeable problem to overcome in order to achieve that purpose if the presence of noise in the images. These images are captured at extremely high frequencies so the exposure time of each frame is really short, of the order of miliseconds.

As the aim on the videos is to record the scene during a relatively large time span it is necessary to use a limited electron density in the incident beam in order to avoid damaging the sample. This fact combined with the short exposure time of each image and the high spatial resolution lead to a very low number of expected electrons per pixel in the resulting image, of the order of magnitude of $O(10^{-1}) - O(10^{1})$ electrons per pixel.

Each travelling electron will, according to quantum mechanics and statistical physics, have a certain probability of reaching the detector after going through the sample, which will vary depending on which region of the sample the electron is traveling through. Therefore physicists can build simulations like the one in Figure 14 using physical models, where a 3D structure is designed and a simulated electron beam is propagated through it. This way we obtain an expectancy of the number of electrons that should be received in each pixel.

When these images are captured with real TEM devices, the result is not this one. As said before, the image displayed in Figure 14 is a computational simulation that shows as the expected number of electrons at each pixel, but as the scenario we have when we capture an image is a detector collecting electrons during a fixed amount of time (the exposure time of the image) and outputting the counting of these received electrons, we can easily identify that the process underlying is a Poisson process.

4.2 Noise in TEM images

We have just stated that the images that we get in our detectors are a Poisson realization of the underlying structure. In fact, we can check that all the requisites of the Poisson distribution stated in 2.3:

- *k is the number of times an event occurs in an interval and k can take values 0, 1, 2, etc.* Here our k variable is the number of received electrons.

- *Events occur independently. That is, the occurrence of one event does not affect the probability that a second event will occur.* The electron beam current in a TEM can be as high as $\sim 0.1 - 1\mu A$, which corresponds to about 10^{12} electrons passing through the specimen plane. These electrons use to be accelerated to carry a energy of $\sim 100keV$ energy, implying that they travel at about $0.5c$ (actually $\sim 1.6 \cdot 10^{8} m/s$), so they are separated by $\sim 1.6mm$ in average. This means that there is never more than one electron in the specimen at the same time, nor in the detector. This distance between electrons ensures that their impacts are completely independent since they are too far from each other to interact in a noticeable way.

- *The average rate at which events occur is independent of any occurrences.* As we stated before, the rate at which events (electron detection) occur only depends on the intensity of the electron beam and the inner structure of the sample. Therefore, once the experiment is set these rates remain constant.

- *Two events cannot occur at exactly the same instant.* As stated before, the distance between electrons in the beam is large enough to ensure that it is virtually impossible for a detector to receive the impact of two electrons at the same time.

Therefore, if the expected number of particles received in the i-th pixel is $x[i]$, the value that we will receive from the detector will follow a distribution $Poiss(x[i])$ leading to noisy images like the one displayed in Figure 15.

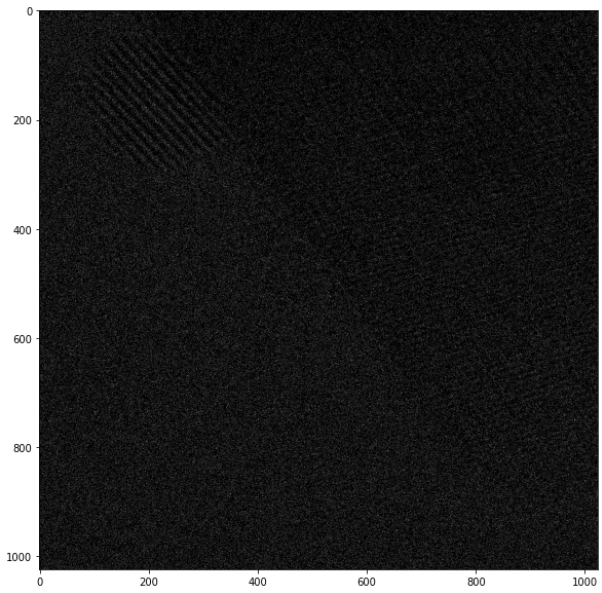

Figure 15: Real noisy TEM image

The removal of this noise is crucial for the propose of catalysis process study since it represents the first and most noticeable barrier before developing any further study. That is the general scope of this project; to make a contribution on the field of scientific image denoising.

From a mathematical point of view image denoising is an Inverse Problem, since there are several clean images that could lead to each noisy observation due to the randomness of the noise, which configures a non bijective function, making it harder to undo the process.

Therefore, there is not a unique and directly applicable solution but a bunch of approaches that could be used. This amount of denoising techniques means another problem due to the lack of a clear criteria to chose which one of these approaches is the best for each case.

Among the state of the art approaches to image denoising listed in Section 3 only a few have been applied to scientific images for two main reasons; the fact that there is not a clear commercial interest to do so and the lack of clean data to train supervised CNNs, which represent the topmost state of the art approach.

This lack of clean data comes from the fact that there is no way to collect a real noise-less image but to use long times of exposure or collecting several images and averaging them to get an estimation. This only option becomes impossible when the system captured in the image is not in a steady state but evolves all the time or when the method of image generation is too aggressive to repeat it or do it during a larger time span, like for X ray imaging where you

cannot administer the patient that much radiation.

Therefore the only way to apply supervised approaches to the problem is by using synthetic images like the one in Figure 14 and adding Poisson noise to them to feed the network during the training process. This is a good solution but has problems of generalization since it is very dependent on the shapes represented on the training set. Then, in order to have a good denoising of an image it is usually necessary to generate synthetic data similar to the target one and use it to train a network for that specific case, which isnot a practical solution.

For that reason the appearance of unsupervised approaches to the denoising field of study represent a turning point in the case of scientific imaging. Recently UDVD 3.2.5 has been successfully applied to TEM images (Mohan et al., 2020) as well as GainTuning 3.2.7. These architectures provide us with a couple of good tools for image denoising but many other of the methods listed in Section 3 could be applied for the purpose.

Among these methods, almost all based on the same idea introduced in Noise2Noise, there are many that only represent small variations of some other. Since UDVD is based on the idea introduced in Noise2Self, which seems to be the one that leads to the best performance, we considered that no unsupervised method previous to this one would be relevant for the aim of the project. Neighbor2Neighbor is, nevertheless, a more recent method that seems worth to take into account.

4.3 Training of general models

As mentioned before, the current approaches for TEM image denoising using Deep Learning techniques have a lack in the generalization; the supervised approaches need to be trained on simulated data similar to the target one, which usually implies training a specific model for each case, and even though the unsupervised approaches can be trained in any data, the standard procedure is to train them on the same data to be denoised each time as the similarity between different datasets is relative.

This lack of generalization is basically caused by the limited amount of images the model learns from during the training process. If a model was trained in a bigger dataset it would be able to learn more general and diverse features in order to perform well in a larger variety of images.

This general models training was prohibitive before the irruption of unsupervised methods since the amount of simulated images available was neither large nor diverse enough, but these last methods enable us to train directly on the noisy data, from which physics groups have a large historic collection.

For that reason, the main product generated during this project is a set of models trained with unsupervised techniques on a large dataset provided by the Cozier Research Group consisting on more than 82,393 real TEM video frames captured during their experiments in summer 2021.

These models may not generalize directly well to all the images on inference but they will have the advantage of having learnt a more general set of features for the denoising process so they should be a good enough base model for a fine tuning process that should drastically reduce the time required for image denoising actually, since nowadays it involves training a whole new model from scratch each time.

4.4 Method comparison metrics

The first and principal purpose of this project is, then, to build a set of Deep Learning based denoising models to be applied on the TEM data that the Cozier Research Group may produce in the future so that they can have a clearer idea of what they are observing in each of the samples they work with.

The only remaining unknown in the models application pipeline is the choice of the best one to apply for each case. For this purpose the standard method is to use the usual noise removal metrics as PSNR, SSIM or the MSE between the noisy and clean images, but that's not applicable here due to the lack of clean data.

Instead, unsupervised metrics should be applied in order to estimate the quality of the denoised image by comparing it with the noisy original observation only.

There is a lack of metrics for that purpose in the literature. The only relevant one is the Stein's unbiased risk estimator described in Section 3.3, which is basically a version of MSE that includes a correction term. This result may be useful in some cases, but for this particular data it may not be the most appropriate since it is based in a Gaussian noise assumption and the noise present in the TEM data is Poisson, as described in Section 4.2.

This leads us to the definition of the second and last purpose of this project, which is the design and evaluation of unsupervised metrics for their application on model discrimination when working in an unsupervised framework. That way we should be able to choose the best model among the generated ones or, if there is not a clearly outstanding option, provide future users with a tool to discriminate the different models' outputs and keep the best one using a criteria more reliable than human perception.

5 Methodology

We will divide the implementation of the solution to each of the previously stated challenges into two main tasks: the model implementation and training and the metric discussion. Inside each of these challenges we will define tasks to fulfill in order to keep track of the project progress.

5.1 Model implementation

5.1.1 Research of the state of the art

The first step in almost any project is to perform some intensive research on the literature to discover what's the state of the art in the field of study and also to find out what are the techniques other researchers have come up with for similar problems.

In our case the field of study is unsupervised denoising and the most relevant developed techniques of the last decade are those listed in Section 3. Almost all the techniques listed there are based on a previous one, for example UDVD is a extension of Self2Self for videos or Neighbor2Neighbor is based on Noise2Noise considering cases where there are no image pairs available and we work on a single image.

In this sense, Figure 16 shows the evolution of these methods and how each is based on an idea of some other. As each method has, generally, outperformed the previous one, we discard all but the leafs of the tree diagram in Figure 16.

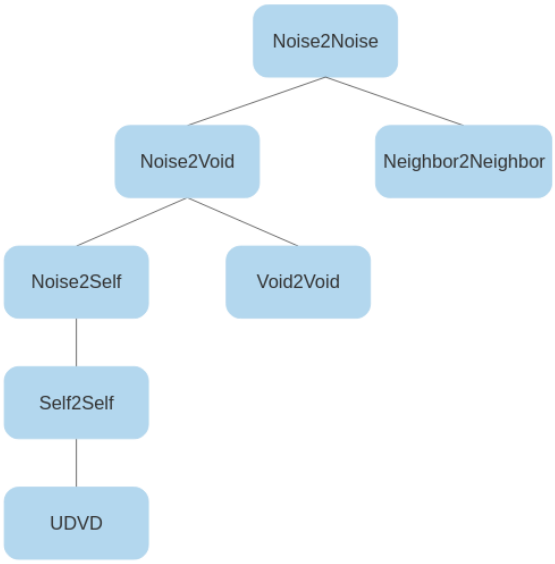

Figure 16: Evolution of unsupervised denoising techniques

That selection leaves us with UDVD, Void2Void and Neighbor2Neighbor. As Void2Void is clearly outperformed by the other two we also discard it from our models choice, replacing it for a Noise2Self model. We make that choice in order to see the influence of including the neighbor frames in the BlindSpot strategy as in UDVD or excluding them as in Noise2Self.

That leaves us with the following models to pre-train on the large dataset from the Cozier Research Group:

- Noise2Self
- UDVD
- Neighbor2Neighbor

5.1.2 Model variations

We also consider some alterations of the current models in order to explore further options. As we have already decided to include Noise2Self in order to check the influence of neighbor frames in the performance of the BlindSpot strategy, we purpose an extended version of Neighbor2Neighbor that takes the continuous frames just as UDVD does and name it V-N2N, a short for Video Neighbor2Neighbor. That way both the BlindSpot and the Neighboring pixels strategies are matched in variations.

Another variation we consider is to implement a version of UDVD that takes a further field of view in order to see if the model is able explode better the periodicity of the TEM data. As very big parts of the TEM images present periodicity in different directions due to the fact that they are atomic crystalline networks, the network tends to use pixels not only from the same region in the image but from the similar regions that are present every certain spatial period when denoising each pixel.

This is something that can be easily checked by computing the Jacobian of the network with respect to a given pixel of the denoised image. In the example of Figure 17 we observe how two concrete pixels from the output have been estimated and we clearly see that not only values from the same region have been used, but also some from neighboring regions.

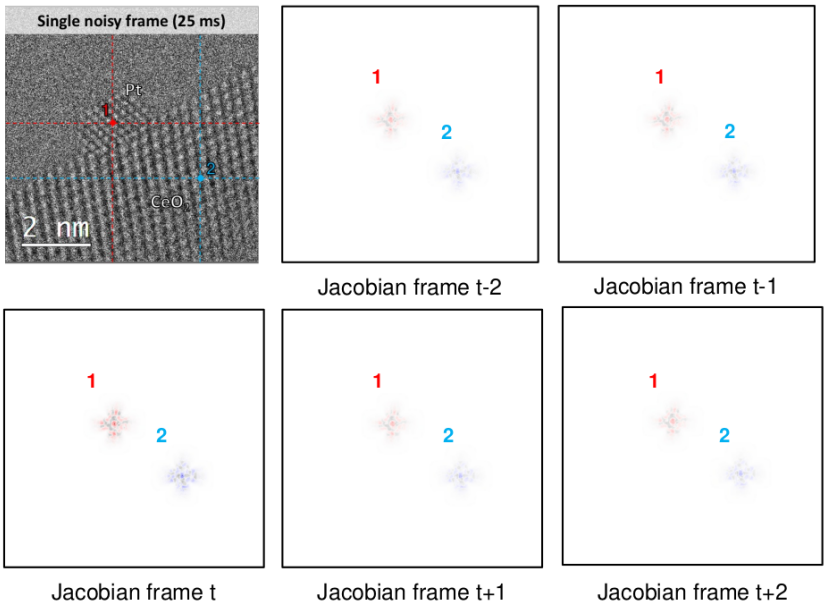

Figure 17: Jacobian of UDVD's output with respect to two different pixels, in blue and red.

In order to explode this property further we decide to increment the region taken into account to compute each pixel. To do so we extend the field of view of UDVD by adding more convolutional and padding layers since each padding layer doubles the field of view size and each convolutional layer extends it some pixels. We name the resulting model as W-UDVD standing for Wide Unsupervised Deep Video Denoising.

That's a modification we don't consider necessary in the case of Neighbor2Neighbor since it has a field of view similar to UDVD but the images are always downsampled by a factor of 2 during the training loop, so the resultant effective field of view of the network is already bigger.

That leaves us with the following models to pre-train on the large dataset from the Cozier Research Group:

- Noise2Self
- UDVD
- W-UDVD
- Neighbor2Neighbor
- V-N2N

5.1.3 Loss functions

In combination with the different networks to be tested we can also explore the different loss functions that can be used. Since all our models are trained using the idea stated in Section 3.2.1, which state that the network will predict the expected value of the pixel if it does not see the true one, we need loss functions that become minimum at the expected value.

The first, typical and almost only choice available in the literature is the Mean Squared Error loss, which is the equivalent as the minimum MSE method for estimator deduction. We therefore train all the models using MSE as a baseline.

In addition, we use the Var_{Err} metric described in Section 5.2.1 to check if it leads to any difference with respect to the MSE trained models' performance. As the noise in our data is Gaussian and the Var_{Err} metric is a continuous, differentiable function that has its minimum on the expected value of a Poisson distribution, we consider it suitable to be a good Loss Function for this problem.

That leaves us with the following losses to try on the process of model training:

- MSE
- Var_{Err}

5.2 Metric design

We will approach the Validation Procedure design by splitting it in two parts. First we will address the process of the Metrics design, where different functions will be defined in order to compare a noisy n and a denoised d image and provide a numerical output.

Secondly, in the next section, different Comparison Techniques will be defined inspired in the ideas of some unsupervised denoising papers. The motivation behind these techniques will be to ensure that, when comparing n and $d(n)$, $d(n)$ is independent from n, or at least from the n we compare it with. By ensuring this we get rid of the bias our metrics would acquire otherwise, for the denoising functions will always compute the denoised image that maximizes some likelyhood criterion based on the noisy input.

The validation procedure design will be addressed by purposing a set of different metrics inspired in different properties of the noisy data plus two unbiased estimators with their corresponding proofs. The former set of metrics will be explored using brute force in order to find the one that works best, whereas the unbiased estimators will be analyzed apart.

5.2.1 Proposed metrics

The first metric to be taken into account is the MSE between the noisy and denoised images. This metric is obviously promising since, as we have already stated in the Introduction, for any noise with $\mathbb{E}[n] = 0$:

$$\mathbb{E}||f(x) - x||^2 = \mathbb{E}||f(x) - c||^2 + \mathbb{E}||n||^2$$

As $\mathbb{E}||n||^2$ is a constant of the noisy image, applying MSE should be equivalent the mean squared error between the denoised and the clean images. We also consider taking into account PSNR but discard it since is proportional to MSE. Nevertheless, we add to our set SSIM in an unsupervised way. The legitimacy of this metric as an estimator of the denoising quality in an unsupervised way is not that straightforward.

We also notice the fact that MSE is highly sensitive to the value range of the data, or more specifically the noise variance. As in the Poisson distribution the noise MSE is linearly related to the λ value of the distribution, which is the true clean value of the pixel, the MSE value will be highly influenced by the image content, which affects to its performance as a basic evaluation metric.

This problem is better understood by means of an example; let X_1 be a clean image of low intensity content and let X_2 be another one with high intensity values. Let Y_1 and Y_2 be their noisy realizations. Finally let f_1 be a good denoiser and f_2 a bad one. Due to the intensity differences it is easy to see that $|f_2(Y_1) - X_1|^2 < |f_1(Y_2) - X_2|^2$ is a highly probable scenario despite f_1 being a better denoiser than f_2.

For that reason we define an additional MSE version where we normalize by the λ value of each pixel:

$$\text{MSE}_{Norm} = \frac{1}{N}\sum_{i=0}^{N}\left(\frac{(n-c)^2}{c}\right)^2$$

Where n are the noisy values of the pixels and c are the clean ones. It is direct to see that the expected value of this metric is 1 under the Poisson distribution due to the equality of the expected value and the variance of this distribution.

We also design a new metric and name it Var_{Err} based on the same idea; As stated in Section 2.3 and mentioned in the previous paragraph one of the peculiarities of the Poisson distribution is the fact that both its variance and expected value are the distribution parameter λ. Therefore the squared error (SE) for each pixel should be a random variable centred on the clean value too. Then we define

$$\text{Var}_{Err}(n, c) = \frac{1}{N}\sum_{i=0}^{N}((n-c)^2 - c)^2$$

Which is a continuous and differentiable function that should show a minimum when the clean image c is the correct underlying image behind the noisy n as we are assuming. We will apply this metric to compare the noisy image with the denoised one instead of the clean, but can do some exploration on its properties when we are using the true clean one.

The expected value of this metric can be easily obtained from its definition:

$$\mathbb{E}\{Var_{Err}\} = \mathbb{E}\left\{((x-\lambda)^2 - \lambda)^2\right\}$$

By taking into account the first moments of the Poisson distribution, which come from the general result

$$m_k = \sum_{i=0}^{k} \lambda^i \left\{ {k \atop i} \right\}$$

where the braces denote Stirling numbers of the second kind. We can then get the first four moments

- $\mathbb{E}\{x^1\} = \lambda$
- $\mathbb{E}\{x^2\} = \lambda + \lambda^2$
- $\mathbb{E}\{x^3\} = \lambda + 3\lambda^2 + \lambda^3$
- $\mathbb{E}\{x^4\} = \lambda + 7\lambda^2 + 6\lambda^3 + \lambda^4$

Then doing some algebra it is easily to prove that our new metric's expected value is:

$$\mathbb{E}\{Var_{Err}\} = \mathbb{E}\left\{((x-\lambda)^2 - \lambda)^2\right\} = \lambda + 2\lambda^2$$

Based on this same idea of comparing the SE of each pixel with its denoised value we also define three more metrics. The first one is Mean_{Err}:

$$\text{Mean}_{Err}(n, c) = \frac{1}{N}\left|\sum_{i=0}^{N}[(n-c)^2 - c]\right|$$

Where we measure the absolute value of the mean error between these two values, the SE and the denoised value.

The second one is the already described SSIM metric (2.9) between the clean values and the squared error.

Finally we also compare them using cosine similarity in order to get another possible measure of how similar these values are.

$$CosSim_{Err}(n, c) = \frac{n \cdot c}{|n| \cdot |c|}$$

These last two metrics are based in the statistical property of mean and squared error being the same in the Poisson distribution but they are not statistical similarity measures so their behaviour could be not as good as expected.

Nevertheless as the aim of this part of the project is to perform a brute force exploration among all the possibilities we decide not to discard them beforehand.

5.2.2 Unbiased MSE estimator

We also introduce two unbiased estimators for MSE and SSIM that are not present in the literature in order to test their efficency and publish them if the results are satisfying.

Theorem 1. *Let x_1, x_2 be two independent realizations of a noisy process with the same expected value $c = \mathbb{E}[x_1] = \mathbb{E}[x_2]$ and let the noise be i.i.d. among realizations. Let $f(x)$ be a denoising function. Then:*

$$\text{uMSE} = \mathbb{E}||f(x_1) - x_2||^2 - \frac{\mathbb{E}||x_1 - x_2||^2}{2}$$

Is an unbiased estimator of

$$\text{sMSE} = \mathbb{E}||f(x_1) - c||^2$$

Proof. Let x_1, x_2 be two independent realizations of a noisy process with the same expected value $c = \mathbb{E}[x_1] = \mathbb{E}[x_2]$ and let the noise be i.i.d. among realizations. Then we can note $x_i = c + n_i$ independently of whether the noise is additive or not. It is straightforward to see that n_1 and n_2 are i.i.d. among them, and that both fulfill $\mathbb{E}[n_i] = 0$, for $\mathbb{E}[x_i] = c$. We can consider them two realizations of a same process n. Then:

$$\begin{aligned}\text{uMSE} &= \mathbb{E}||f(x_1) - x_2||^2 = \mathbb{E}||f(x_1) - c - n_2||^2 = \\ &= \mathbb{E}||f(x_1) - c||^2 + \mathbb{E}||n_2||^2 + 2 \cdot \mathbb{E}||f(x_1) - c|| \cdot \cancel{\mathbb{E}||n_2||} = \\ &= \mathbb{E}||f(x_1) - c||^2 + \mathbb{E}||n||^2\end{aligned}$$

$$\begin{aligned}\frac{\mathbb{E}||x_1 - x_2||^2}{2} &= \frac{\mathbb{E}||\cancel{c} + n_1 - \cancel{c} - n_2||^2}{2} = \\ &= \frac{\mathbb{E}||n_1 - n_2||^2}{2} = \frac{\mathbb{E}||n_1||^2 + \mathbb{E}||n_2||^2 + 2\cancel{\mathbb{E}||n_1||} \cdot \cancel{\mathbb{E}||n_2||}}{2} = \\ &= \frac{\mathbb{E}||n||^2 + \mathbb{E}||n||^2}{2} = \mathbb{E}||n||^2\end{aligned}$$

Hence:

$$\text{uMSE} = \mathbb{E}||f(x_1) - x_2||^2 - \frac{\mathbb{E}||x_1 - x_2||^2}{2} = \mathbb{E}||f(x_1) - c||^2 + \mathbb{E}||n||^2 - \mathbb{E}||n||^2 = \mathbb{E}||f(x_1) - c||^2$$

□

5.2.3 Unbiased SSIM estimator

Later in the project development this estimator is proven to be wrongly designed since its proof is not correct. Nevertheless it is relevant to know how the proof was stated in a first moment in order to understand the conclusions extracted from that.

Definition Let X, Y be two images, the SSIM(x, y) index is calculated on various windows of these images. The measure between two windows x and y of common size $N \times N$ is:

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}$$

where:

- μ_x is the average of x;
- μ_y the average of y;
- σ_x^2 the variance of x;
- σ_y^2 the variance of y;
- σ_{xy} the covariance of x and y;
- $c_1 = (k_1 L)^2$, $c_2 = (k_2 L)^2$ two variables to stabilize the division with weak denominator;

- L the dynamic range of the pixel-values (typically this is $2^{\#bits\ per\ pixel} - 1$);
- $k_1 = 0.01$ and $k_2 = 0.03$ by default.

Definition Let X, Y, Z be three images, the uSSIM(x, y, z) index is calculated on various windows of these images. The measure between three windows x y of and z common size $N \times N$ is:

$$\text{uSSIM}(x, y, z) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 - P_z + c_2)}$$

where P_z is half the power of z, $\mathbb{E}||z||^2/2$.

Theorem 2. *Let x_1, x_2 be two independent realizations of a noisy process with the same expected value $c = \mathbb{E}[x_1] = \mathbb{E}[x_2]$ and let the noise be i.i.d. among realizations. Let $f(x)$ be a denoising function. Then uSSIM$(f(x_1), x_2, x_1 - x_2)$ is an unbiased estimator of SSIM$(f(x_1), c)$.*

Proof. Let us start defining some notation for simplicity. Let $d = f(x_1)$ be the denoised version of x_1 and let σ_d, μ_d be its statistics in the SSIM formula. Let σ_c, μ_c be the equivalent for the clean image c and let σ_x, μ_x be them for the noisy image x_2. Let m be $x_1 - x_2$. We are then trying to proof the following equality:

$$\text{uSSIM}(d, x, m) = \frac{(2\mu_d\mu_x + c_1)(2\sigma_{dx} + c_2)}{(\mu_d^2 + \mu_x^2 + c_1)(\sigma_d^2 + \sigma_x^2 - P_m + c_2)} = \frac{(2\mu_d\mu_c + c_1)(2\sigma_{dc} + c_2)}{(\mu_d^2 + \mu_c^2 + c_1)(\sigma_d^2 + \sigma_c^2 + c_2)} = \text{SSIM}(d, c)$$

We can then proof the equality element-wise by proving the following set of equalities:

1. $\mu_d = \mu_d$
2. $\mu_x = \mu_c$
3. $\sigma_{dx} = \sigma_{dc}$
4. $\sigma_d = \sigma_d$
5. $\sigma_x - P_m = \sigma_c$

Obviously 1 and 4 are direct. Let us address the others:

1. $\mu_d = \mu_d$
2. $\mu_x = \mathbb{E}[x] = \mathbb{E}[c + n] = \mathbb{E}[c] + \cancel{\mathbb{E}[n]} = \mathbb{E}[c] = \mu_c$
3. $\sigma_{dx} =$
$$= \mathbb{E}[(d - \mu_d)(x - \mu_x)] =$$
$$= \mathbb{E}[(d - \mu_d)(c + n - \mu_c)] =$$
$$= \mathbb{E}[(d - \mu_d)(c - \mu_c)] + \mathbb{E}[(d - \mu_d)n] =$$
$$= \mathbb{E}[(d - \mu_d)(c - \mu_c)] + \mathbb{E}[(d - \mu_d)]\cancel{\mathbb{E}[n]} = \mathbb{E}[(d - \mu_d)(c - \mu_c)] = \sigma_{dc}$$

4. $\sigma_d = \sigma_d$

5. $\sigma_x - P_m =$

$$= \mathbb{E}[(x-\mu_x)^2] - \mathbb{E}\left[\frac{(x_1-x_2)^2}{2}\right] =$$

$$= \mathbb{E}[(c+n-\mu_c)^2] - \mathbb{E}\left[\frac{(\cancel{c}+n_1-\cancel{c}-n_2)^2}{2}\right] =$$

$$= \mathbb{E}[(c-\mu_c)^2 + n^2 + 2(c-\mu_c)\cdot n] - \mathbb{E}\left[\frac{n_1^2+n_2^2-2n_1n_2}{2}\right] =$$

$$= \mathbb{E}[(c-\mu_c)^2] + \mathbb{E}[n^2] + 2\mathbb{E}[c-\mu_c]\cdot \cancel{\mathbb{E}[n]} - \frac{\mathbb{E}[n_1^2]+\mathbb{E}[n_2^2]-2\cancel{\mathbb{E}[n_1]}\cdot\cancel{\mathbb{E}[n_2]}}{2}] =$$

$$= \mathbb{E}[(c-\mu_c)^2] + \mathbb{E}[n^2] - \frac{\cancel{2}\mathbb{E}[n^2]}{\cancel{2}}] =$$

$$= \mathbb{E}[(c-\mu_c)^2] + \cancel{\mathbb{E}[n^2]} - \cancel{\mathbb{E}[n^2]} = \mathbb{E}[(c-\mu_c)^2] = \sigma_c$$

□

5.2.4 Metrics summary

We therefore define the following metrics:

- MSE
- SSIM
- Var_{Err}
- Mean_{Err}
- $CosSim_{Err}$
- SSIM_{Err}
- uMSE
- uSSIM

5.2.5 Comparison Techniques

The second half of the metric design is to decide the comparison techniques to be applied. That is what pixels are we using when computing the metrics. This discussion comes from a bias detected in our initial approaches:

Let y be a noisy image and let $f(y)$ be the denoised version obtained by means of a given denoising technique f. Then the i-th pixel of $f(y)$, $f(y)_i$ has a value that is a weighted combination of a set of values $\{Y_j\}_{j\in J}$ Where J is the neighborhood of i and generally $i \in J$.

This formal statement is equivalent to saying that each denoised value is a combination of the

noisy pixels that surround its position and the same noisy pixel itself.

The direct conclusion we can extract from this is that, if $i \in J$, there will exist an evident correlation between $f(y)_i$ and y_i. This fact makes any pixel-wise computed metric $M(y, f(y))$ biased since we are trying to compare each denoised pixel with its original noisy version, and the denoised one has been computed in order to maximize the likelihood of it being the underlying value behind the noisy, according to some livelihood criterion.

The conclusion we extract from this idea is that, if $i \in J$, we can not validate a method using metrics that compare directly y and $f(y)$ pixel-wise. We therefore purpose the following techniques to avoid this bias:

- Pixel masking.

  Our first option is to generate a masked version of the y image $\hat{y}$ where the 10% of the pixels have been masked. To do so we chose randomly a set of pixels B and, for each of them, we substitute their value by the median of the pixels surrounding it.

  Once this masked version is ready we apply our denoising technique f to obtain $f(\hat{y})$ and compute our metric M only on the positions where masking has been applied, comparing the denoised pixels with the unmasked true values through our metric $M(y_B, f(\hat{y})_B)$.

  This way the values we are comparing are the noisy pixels y_B and the denoised values that we have in the same positions $f(\hat{y})_B$, which have been computed without using any of the values y_B in any moment.

  This way we get rid of the relation between noisy and denoised values, but the metric is computed only on the 10% of the pixels, which makes it less robust.

  Also, this technique can only be used for pixel-wise metrics, which excludes SSIM since it uses context information.

- BlindSpot architectures.

  The second and easiest option is to use BlindSpot architectures, which ensure us that the denoised value of a given pixel $f(y)_i$ has no relation at all with the noisy value of the same pixel y_i because of the BlindSpot architecture.

  This option ensures us that we use all the pixels but is very restrictive since it is based on the usage of only BlindSpot architectures, so it does not really represent a method that can generalize to any denoiser in order to compute its efficiency.

  Also, this independence between the noisy and denoised pixels only works pixel-wise, for any metric applied on a set of pixels that mixes different positions will still be affected by the bias.

- Neighbor sampling.

  From the Neighbor2Neighbor denoising technique we extract the idea of subsampling the images to create a set of four totally independent realizations of the same process, despite it implies losing half of the resolution.

Using this sampling technique we can directly extract two different images from a single sample (y_1, y_2) and then apply our metric as $M(y_1, f(y_2))$ ensuring that we compare two completely independent sets of pixels.

This method seems a very promising option but has a nontrivial problem since the denoising technique applied to the data may not perform the same way when we sample it to half of the original resolution.

That's easy to see in the case of Deep Learning techniques, where the model learns to process what it sees during the training loop. Then, if a model has only been trained with images of a specific resolution, and given how the models seem to explode periodicity in the case of TEM images, it would be expectable to get a poorer performance on images with a different resolution.

- Successive frames comparison

  The last option explodes the steadiness of most of the TEM videos, where no notorious changes occur between neighboring frames. We then purpose a technique consisting in validating the denoised output of a frame $y[t]$ by comparing it with its temporal neighbor $y[t+1]$, ensuring that the compared data is independent.

  This method has several lacks. The first one is the fact that, since it is only applicable to videos due to the need of successive frames, is can not be a reference metric for general image denoising problems since most of them try to denoise images and not videos.

  We are also assuming independence between $f(y)[t]$ and $y[t+1]$ which is not true in the case of UDVD or V-N2N since they consider a set of successive frames to denoise the central one.

  Finally we are assuming that $y[t]$ and $y[t+1]$ can be considered as two realizations of the same process, implying that so important changes occur between successive frames. That can be considered true in most of the TEM data we have in the current dataset but generalizes very poorly since the samples that can be analyzed at atomic scale can also be very chaotic and rapidly changing.

5.2.6 Metrics and techniques summary

We are finally defining the intersection between metrics and techniques to be explored. Despite the fact that we could address all the possible and compatible combinations, 9 metrics and 4 comparison techniques lead to a very big set of combinations.

We will therefore eliminate some of them, those that seem less relevant or the incompatible pairs.

	Direct comparison	Pixel masking	Neighbor sampling	Successive frames
MSE	yes	yes	yes	yes
MSE$_{Norm}$	yes	yes	yes	yes
SSIM	yes	no	yes	no
Var$_{Err}$	yes	yes	yes	yes
Mean$_{Err}$	yes	yes	yes	yes
CosSim$_{Err}$	yes	no	yes	no
SSIM$_{Err}$	yes	no	yes	no
uMSE$_{Err}$	no	yes	yes	yes
uSSIM$_{Err}$	no	no	yes	yes

Figure 18: Metrics summary.

Notice that not all the combinations of metrics and techniques are covered because some metrics are simply being initially explored and some are incompatible. That's the case of the SSIM based metrics and the pixel masking technique.

Notice also that both unbiased estimators are incompatible with performing direct comparison since they require two realizations of the pixels to be compared.

In addition we also consider using Stein's unbiased risk estimator as a regularized MSE technique. It will only be applied in the Direct comparison technique as a first exploration and if the results are promising other techniques will be considered.

This extra method is only considered to cover the lack of a regularized MSE metric but is not expected to perform well since the Gaussian assumption it is based on is not fulfilled by the Poisson noise present in the TEM data.

5.3 Metric evaluation

The idea of the metric evaluation is to compare the different metrics that we purpose, which use noisy and denoised images, with some well known metrics that use the clean and the denoised ones.

Currently MSE, PSNR and SSIM are three metrics that are considered the reference ones to compare image quality. This way we can compute their values when comparing the denoised image that we get as result of applying a given denoiser to a noisy sample and the clean original image we had before adding any noise. These values are the most reliable metrics that we can use to measure how good the denoised image is, since when comparing it with the true clean image we get a direct score of how similar out result is to the optimal one.

These three metrics plus the Normalized MSE defined in Section 5.2.1, also applied to compare the denoised and clean images, are our reference metrics that enable us to know how good a denoised image actually is.

On the other hand we have the huge set of different metrics we have defined previously, which we want to test and validate.

In order to perform a good comparison between the noisy-denoised metrics and the noisy-clean ones we will need several trios of noisy-denoised-clean images where, ideally, each clean image

is paired with its noisy version but several denoised options exist. This way we will be able to determine if out metrics are able to tell us which of them is the best.

5.3.1 Evaluation data

In order to evaluate the performance of our proposed metrics and discriminate those that perform the best we will start by building a dataset of benchmark clean images using the most commonly used data among the denoising literature. Once this is done we will get their noisy equivalents by adding computer simulated noise.

In our final selection we will have BSDS300 (Hou, Yuille, & Koch, 2013) and KODAK(Toderici et al., 2017), both for natural images captured using digital cameras, MEDPIX (Buendía, Gayoso-Cabada, & Sierra, 2018) containing a set of thorax X-Rays, PolyU (Zhong, Yang, & Du, 2018) containing fingerprints and finally Set14, a small set with the typical benchmark image processing examples such as Camera Man or Pamela.

We also include DAVIS (Pont-Tuset et al., 2017) for natural videos and some simulated TEM videos in order to be able to test the successive frames technique.

For convenience and also due to the fact that the TEM images this project is based on are gray-scale, we transform all the RGB images in the natural images or videos in these datasets into gray-scale and store them in this format. This way we will ensure that we always work with one-channel images.

5.3.2 Noise addition

As we stated in previous sections, the noisy images are generated by adding Gaussian and Poisson noise to the clean data. In order to obtain denoised images that cover a wider range of qualities, we define different noise levels for each of the noise distributions.

Taking into account that all the images are normalized so that their value range is $[0, 1]$ with the exception of the Nanoparticles data where the range was already around $[0, 1.5]$, we decide to add Gaussian noise with the following σ values: $\{0.005, 0.01, 0.025, 0.05, 0.075, 0.1, 0.25\}$.

The case of the Poisson noise is not that simple because it is not an additive process but this can be easily solved by defining a parameter μ and generating the noise as $Poisson(x \cdot \mu)/\mu$ which sets the standard deviation of the noise to be λ/μ instead of λ without altering the mean value. We then set the following values for μ: $\{1, 2.5, 5, 10, 50, 100, 200\}$.

5.3.3 Denoiser selection

We continue adding variety to our data by building a set of denoisers that will enable us to get different denoised versions of a given noisy image in order to compare the values of our tested metrics and the reference ones. The chosen denoiser options are:

- Wavelet filtering
- Wiener filter of radius 5 pixels
- Wiener filter of radius 10 pixels

- BlindSpot CNN
- Supervised CNN

Several versions of the CNNs are trained in order to apply them to the data later. Our dataset gets divided in four different subsets where different networks are trained: MEDPIX, PolyU, Nanoparticle videos and Natural images/videos. For each subset of images we train two different CNNs of each kind, one using Poisson noise and the other one using Gaussian, each noise applied with different intensities mixed.

This means that 16 models are trained in total (4 subsets * 2 types of noise * 2 CNN architectures) in order to have a specific denoiser for each kind of data.

5.3.4 Data structure

The previously stated process of noise addition and image denoising can be summarized to the fact that we are using 2 types of noise, each with 7 different intensities, and we are denoising using 5 different methods. This implies that for each clean image we get 14 different noisy ones and 70 denoised, from which one half has Gaussian noise and the other one has Poisson.

We then compute the different metrics, both the reference and the tested ones, on each of the denoised images and store the results in a table. This way we end up building a table composed of 489,160 rows that store the name of the original image from which each row comes, the noise type and intensity applied and the values of the different metrics, which add up to 30 columns.

5.3.5 Metrics visual comparison

Once all the metrics have been computed and properly stored in a structured scheme we can start comparing the metrics and their performance. The main objective is to find the metric or set of them that correlate the best with the reference ones. In a formal way:

Let m_r be a reference metric, such as SSIM, between denoised and clean images, and let's assume for convenience that its value is directly proportional to the denoised image quality, equivalent to say that it is proportional to the image similarity between the compared images. Let m_p be a proposed metric that compares the noisy and denoised ones, which value is also proportional to the image quality. Then let x_g be a given observation, a row of our scheme, where the denoised result is pretty good, and let x_b be another one where the denoised result is worse, that is it doesn't look that much as the clean image.

Then it would be expectable that $m_r(x_g) > m_r(x_b)$ since m_b's value is proportional to the denoised image quality. If our proposed metric is valid it should lead us to the same conclusion, then we should also see that $m_p(x_g) > m_p(x_b)$.

We can summarize this by stating that our metric will be working well if $\forall(x_1, x_2)$, $m_p(x_1) > m_p(x_2) \iff m_r(x_1) > m_r(x_2)$, which is equivalent to say that the curve $(m_p(x), m_r(x))$ is non-decreasing. As some of the reference and proposed metrics are not directly proportional to the image quality but inversely, we have to add non-increasing curves to this definition, which compose the set monotonic curves.

To sum up, we can affirm that our metrics will be working well if their plot against our reference metric is a monotonic curve.

This is the main idea around which we will be gravitating during the whole metric testing and validation process. The first way to test it is the visual one. For that reason we build a matrix of scatter plots where we compare each of the metrics we have defined, whether reference or proposed, with all the others. By doing this we not only check the behaviour of some of the proposed metrics when compared with the reference ones but also check some of the relations between different proposed metrics and look for strong correlations among the different Techniques (5.2.5) to apply each of them.

5.3.6 Spearman's rank correlation coefficient

As a visual comparison is not enough for any valid conclusion we need to move to numeric metrics. For that purpose, as the only relevant property to be checked in our data is it monotonicity, we apply the Spearman's rank correlation coefficient (SRCC) introduced in Section 2.10 to each of the metric combinations. As we do not really care about the sign of the monotony of our metrics' relations as long as it exists, we substitute the SRCC by its absolute value in order to focus the attention on the important conclusions.

To ease the evaluation process we keep arranging the metric pairs in a matrix and encode the absolute value of the SRCC using the intensity of the intersection box color. This way the most optimal metrics will catch attention much faster.

Finally we also build a variation of this visualization tool where, instead of computing the SRCC on each pair of metrics using all the images, we compute it per image and average all the results.

That is, for each clean image, we get all its noisy and denoised derivatives and compute the different metrics we want to compare. Then we compute the SRCC only on these images, obtaining a SRCC value per each clean image. Finally, we average all the images SRCC values to get a single one. We name this measure SRCC per Image (SRCCpI).

The motivation behind this variation of the technique is the fact that we suspect some of the metrics can be highly influenced by the image contents, like in the case of MSE where, as we have already explained, a low value can be due to the fact that the denoised image has a good quality of can come from the value range of the clean one, specially when the Poisson noise is considered.

Therefore, in order to remove this possible content influence to the metrics performance, we decide to check whether a given metric is capable of distinguishing the best denoised version of a same image. By doing this the content remains constant among data points, which ensures us that no content dependency is influencing our results.

All the procedures stated in this section will be applied to different selections of the data scheme in order to check the influence of different factors such as the type and intensity of the noise when we apply each of the metrics.

5.4 Model selection

Once all the previous steps have been finished we should have a set of models trained on real noisy data and a metric or set of them that enables us to compare the different models' performances in the lack of clean data.

At this point the only remaining tasks are the testing of the different models' performances on the Test set and the selection of the best one. Once this is done we will train a final version of the model on the whole dataset and save it in order to be used as a basic model to perform fine tuning whenever new data needs to be denoised.

The final step will be the implementation of a user friendly script that enables the Physicists in the Crozier Research Group to denoise data faster without the need of any Deep Learning background and deliver it to them for their testing.

6 Results

6.1 Models

The first task to be addressed in the practical implementation of this project was the model implementation and training. In that sense the following models were trained on the Cozier Research Group dataset:

- Noise2Self
- UDVD
- UDVD with Var_{Err} loss
- W-UDVD
- Neighbor2Neighbor
- V-N2N

Notice that all the networks have been trained using the MSE loss but a UDVD one that has been used as a proof of concept.

Some of the networks, specially those based on the blind spot technique, namely Noise2Self and the UDVD variations, have been trained using the early stopping technique; as these networks tend to overfit on the training data after a certain amount of training loops, we keep track of the validation loss and save the model the this metric achieves its minimum.

This overfitting behaviour is expected since the authors of the different architectures prevent the reader about them, but in the context of the current project it has not been noticed. We can speculate that it may be due to the usage of a bigger dataset, since all the used architectures are proposed to address problems with reduced amounts of data, not only to cover these critical cases but probably because of the lack of a large available dataset at the moment.

Despite the overfitting absence we keep the early stopping strategy to have a more general code due to the fact that this technique does not represent any difference when no overfitting is present, for the minimum of the validation loss tends to occur in the later training steps.

The training conditions and results for the different networks are summarized in the table in Figure 19.

Model training summary				
Architecture	Loss function	Early stopping	Epochs	Validation loss
Self2Self	MSE	Yes	300	46.5
UDVD	MSE	Yes	300	40.5
UDVD	Var_{Err}	Yes	300	39222
W-UDVD	MSE	Yes	300	38.5
N2N	MSE	No	300	46.7
V-N2N	MSE	No	300	45.2

Figure 19: Model training summary.

6.2 Brute force metric evaluation

The metric evaluation has been performed based on groups of three images: a *clean* one, a *noisy* version of the same and a *denoised* obtained by denoising the previous one. To compute the *noisy* images both Gaussian (with σ values $\{0.005, 0.01, 0.025, 0.05, 0.075, 0.1, 0.25\}$) and Poisson (reescaled using μ values $\{1, 2.5, 5, 10, 50, 100, 200\}$) were used, producing several data points for each original *clean* image.

Once all the images have been computed the different metrics are stored in the table introduced in Section 5.3.4 and the different visualizations are generated in order to compare them. All these visualizations can be found in the Annex A, Section 9.

6.2.1 General scatterplot exploration

In a first exploration all the variables and metrics were used in order to display a general picture of the metrics relations (Figure 25) in order to observe the most obvious trends and correlations among them.

In general, we observe linearity between the different Comparison Strategies for a given metric but that is not always the case; we can appreciate how the Comparison Technique that seems to relate the poorest with the others seems to be the usage of neighbor frames, while the one that seems to have a strongest linear behaviour is the subsampling one, specially for the Mean_{Err} and SSIM_{Err} metrics. This poor performance of the neighbor technique is perfectly understandable, for it is the only one that does not use the same data but assume that a different frame is another valid realization of the same process, which does not hold completely for our data judging by the observed performance.

We can also notice how certain metrics have a clear and expected correlation, like the case of MSE and PSNR, but also unexpected in some other cases. That's the case of Mean_{Err} and SSIM_{Err} for example, which show a monotonic relation despite not being related a priori.

Last but not least, it is interesting to see how certain variables, such as Var_{Err} and Mean_{Err} don't show a single linear trend but several ones when compared with variations of themselves or others. As we are displaying all the noise types and denoiser functions together we can theorize that the different branches may have their origin in different generations of the data. The hypothesis of the noise gets invalidated when we check the same plot generated using only the Poisson noisy data (Figure 26) or the purely Poisson (Figure 28), where we keep observing the same trends, but seems to hold for the Gaussian (Figure 27) where only one trend is observed in most of these cases.

6.2.2 General monotony exploration

Moving into the SRCC and SRCCpI values we are able to measure in a more understandable way the correlation we observe among these variables. These two correlation metrics are expected to be enough as we only care about the monotony of the metric pairs' behaviour. It is relevant to point how SRCCpI is the most reliable criteria among both as it measures how well the metrics perform comparisons on the same data, and the comparisons among different models use to be performed evaluating their performances on a common test dataset.

If we consider MSE and SSIM to be the most reliable reference benchmark metrics (i.e. the most reliable way to compare the *denoised* and *clean* images) we can check the general comparison in

Figure 33 and state that the most reliable unsupervised metrics for generic noisy images are, as expected, benchmark ones applied to compare the *noisy* and $denoised$ images. in other words, the metrics that correlate the best with the supervised MSE and SSIM are the unsupervised MSE and SSIM, with the peculiarity that the unsupervised MSE seems to be more related with the supervised SSIM than it is with the supervised MSE. On the other hand, Var_{Err} and $cossim_{Err}$ seem to be the less reliable variables, not only when compared with the benchmarks but also with respect to the other metrics, where the correlation coefficients obtained are all pretty low.

This poor correlation holds for all the kinds of noises in the case of $cossim_{Err}$ while Var_{Err} seems to perform better on Poisson noise, as one could expect since it is based on this distribution assumptions. This can be seen on Figures 30 to 36 by checking the columns corresponging to the different versions of these variables on each subset of data.

For that reason, at this point of the results analysis, $cossim_{Err}$ is eliminated from the set of tested metrics and will nor appear in any further figures. Also, now that the big picture has been analysed in a first approach and in order to focus on the relevant comparisons we discard all the non-relevant metric pairs and will only focus in comparing the benchmark metrics applied in a supervised way with all the metrics applied in an unsupervised way.

6.2.3 Concrete monotony exploration

In figures 45 to 48 we can clearly observe how the benchmark metrics perform in a pretty decent way when applied to compare the *noisy* and *denoised* images and how MSE and SSIM seem to still be the best Metric options, also for this application. We can also conclude that the best Comparison Technique seems to be the subsampling one according to all the realizations.

In addition, we notice how the former affirmation holds for all the data subsets but the pure Poisson one represented in Figure 48, where the `cmse` metric seems to surpass the others. This is the implementation of the MSE_{Norm} Metric. This metric seems to level with the other two remarked ones, usually getting SRCCpI values slightly smaller and surpassing them in this particular case of the pure Poisson noise.

On the other hand, figures 57 to 60 show us how the proposed metrics relate with the reference ones. Keeping in mind how all these metrics are based on the idea of the squared error and mean value equality of the Poisson distribution, which turns into a linearity for the case of the re-scaled Poisson noises, it is not a surprise to observe a poor performance in the case of Gaussian noise when checking Figure 59.

In these figures we can still point Var_{Err} to be the poorest performing Metric and we can also confirm the subsampling Comparison Technique as the best one, despite none of the metrics considered seem to surpass the performance of the previously analysed benchmark ones.

We can therefore assume that the best performance both in the general analysis and in the per noise one is achieved by SSIM and MSE when they are computed using the subsampling approach.

6.3 Unbiased metrics evaluation

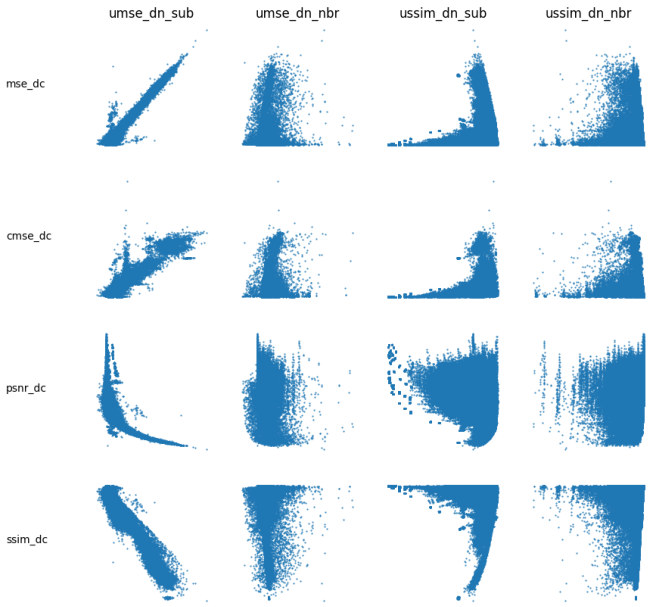

Figure 20: Scatterplot of the unbiased estimators using all the images, noise types and denoisers.

We start the evaluation of the unbiased estimators by exploring the same kind of scatter plots we have used in the previous section. According to Figure 20, it is clear that uMSE is a good estimator when combined with the subsampling method since a single, clearly linear trend with slope 1 is observed when considering all the images together, without need of splitting the data per noise type. On the other hand it does not seem to behave that well when the successive frames comparison technique is applied.

Nevertheless, it is also clear how the uSSIM estimator does not show the correlation that would be expectable from an unbiased estimator. After noticing that the value range of the SSIM and uSSIM is very close to 1, for the denoising methods applied to the data were working well, a different experiment was designed: by taking one of the CNN used for the denoising in the process of generating these figures and disturbing its weights with Gaussian noise we obtain a set of denoisers with poor performance that can be used to check the behaviour of uMSE and uSSIM on a wider range of cases.

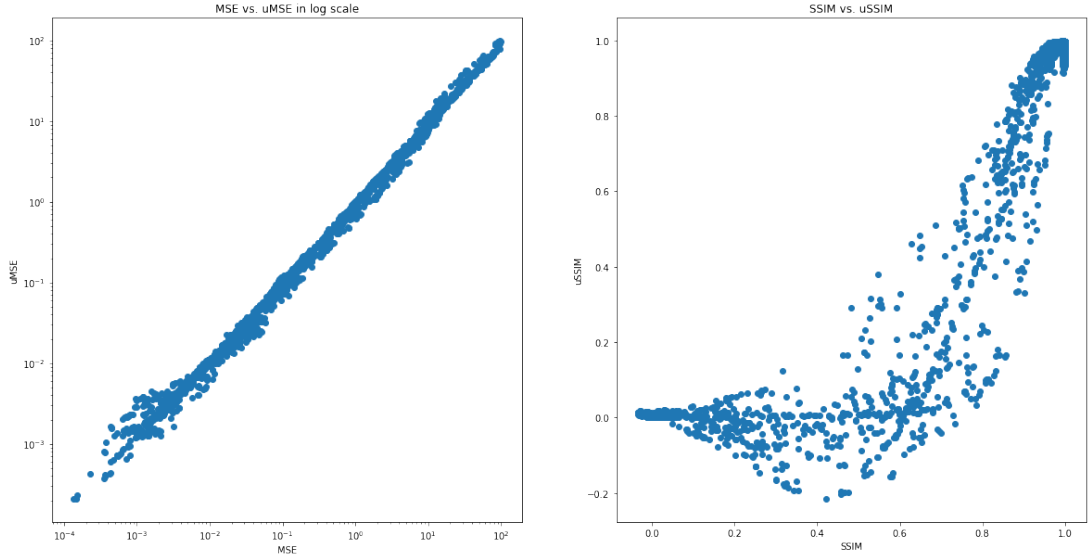

Figure 21: Scatterplot of the unbiased estimators using all the images.

We can now appreciate in Figure 21 how the behaviour of uMSE is the best that we could have expected whereas uSSIM does not behave that well. We theorize two possible reasons for that behaviour:

- The fact that uSSIM is computed using a 11×11 sliding window may not provide enough samples for the Law of the Great Numbers to apply, so the σ_c estimation is not correct.
- The different elements in the SSIM formula could not be addressed independently in the estimator deduction of Section 5.2.3.

As the first reason has no solution and no conclusions have been achieved when considering the second one, for the addition of interaction to the SSIM formula makes the problem much more complex, no further knowledge on this problem has been achieved.

Keeping this in mind, and taking into account how uSSIM acieves only a SRCCpI of 0.843 on the global scene whereas the classic SSIM applied to compare noisy and denoised images with the subsampling technique achieved a 0.920 SRCCpI, as can be checked on figure 45, we can conclude that the best choice for a SSIM estimation is this second one.

As it was expectable by checking the different figures, specially the MSE plot of Figure 21, the uMSE metric surpasses all the other metrics considered when trying to estimate the MSE. In terms of the SRCCpI, uMSE achieved a 0.943 average correlation whereas the denoised vs. noisy MSE applied using the subsampling technique only got a 0.842.

For those reasons we end up concluding that uMSE$(denoised, noisy)$ and SSIM$(denoised, noisy)$, both applied using the subsampling technique, are the best criteria to compare different models on a common validation dataset.

6.4 Model comparison

Once the metric exploration has been performed and the best unsupervised metrics have been chosen it is time to use them in order to compare the different models that have been trained on our dataset. For that purpose a common validation dataset is considered and the different models are applied in order to measure their performances. They will be compared using the uMSE$(denoised, noisy)$ and SSIM$(denoised, noisy)$ metrics, where the confidence intervals will be computed in order to get a more general picture.

We start by comparing the average uMSE values obtained for each of the trained models and get the results summarized in Figure 22

Model uMSE comparison summary			
Architecture	Loss function	Epochs	Mean uMSE
Self2Self	MSE	300	1.787635
UDVD	MSE	300	-1.668155
UDVD	Var_{Err}	300	87.057700
W-UDVD	MSE	300	-1.539050
N2N	MSE	300	1.162821
V-N2N	MSE	300	-1.157918

Figure 22: Mean uMSE value for each model.

Notice that many of the uMSE values are negative. That's perfectly comprehensible since the formula of the estimator contains a negative term and indicates that the MSE is close to zero.

Despite the table in Figure 22 is not definitive for the comparison, for a simple average comparison is not enough to conclude that a given model performs better than another one, it is useful to detect how the Var_{Err} loss does not lead to satisfactory denoising.

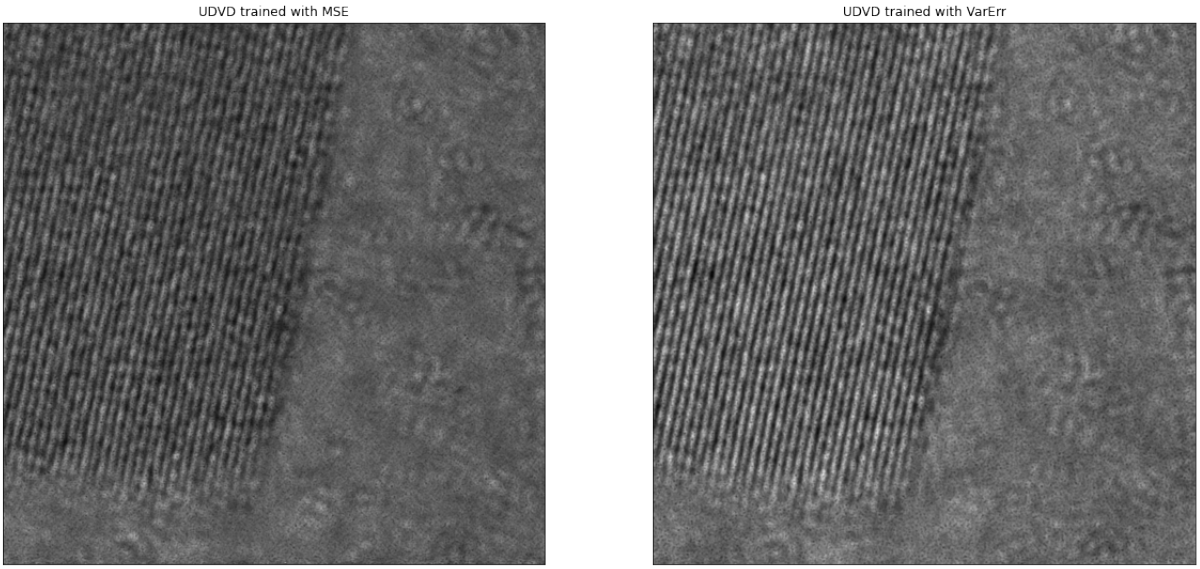

Figure 23: Example of the MSE and Var_{Err} trained networks performance.

Nevertheless, it is interesting to see how it actually performs denoising. In Figure 23 we can appreciate the output of UDVD when trained with MSE and Var_{Err} losses for a common input. The Var_{Err} trained version of UDVD seems to recover the structure but fail on the value range. For the example of Figure 23 we have a value range of approximately $[5, 7]$ for the MSE trained network whereas the Var_{Err} one leads to a range of approximately $[15, 17]$.

The conclusion we can extract from this is that the Var_{Err} loss leads to noise removing plus a value shift. Noticing this and considering how the true clean values of the images are relevant for the physicists at the Crozier Research Group we decide to eliminate completely this variation of UDVD from the following analysis.

We compare then the models using a boxplot in order to visualize the range of values that uMSE and SSIM return on each model. In Figure 24 we can appreciate the results, noticing again how many of the uMSE values lay in the negatives.

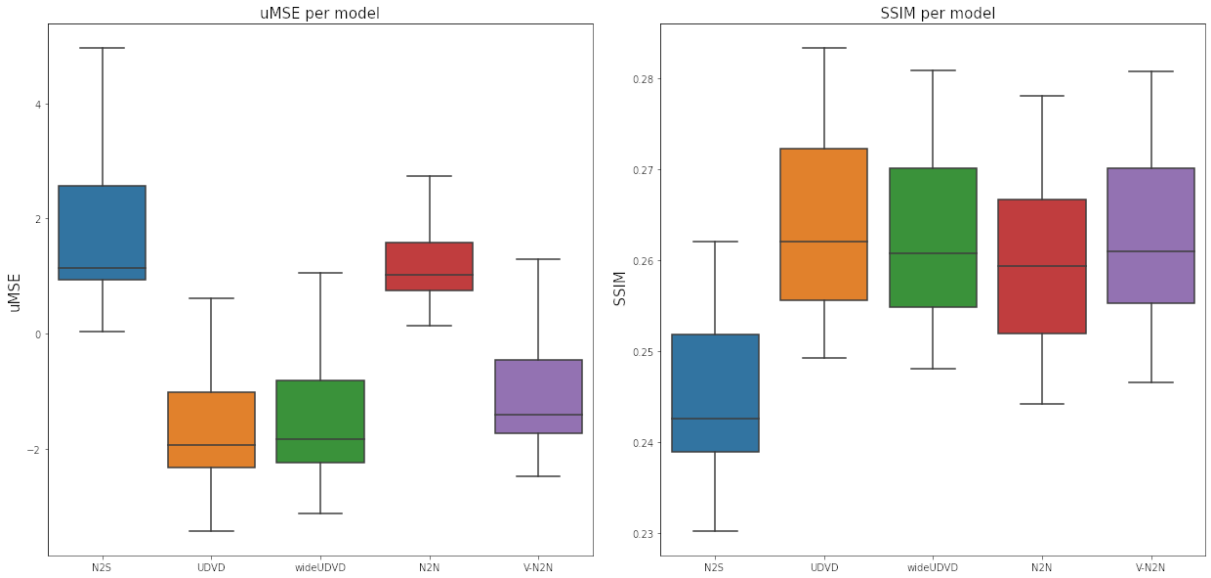

Figure 24: Boxplot of uMSE for each model.

Even though considering a negative MSE estimate may look like a strange thing to do, we consider that these values indicate a really low true MSE value and take them as valid, for the results observed during the uMSE testing and validation give us confidence on the reliability of the uMSE metric.

According to the results displayed in Figure 24 the best model is UDVD closely followed by its version with a wider field of view (wideUDVD) and Video-Neighbor2Neighbor according to both uMSE and SSIM. It is interesting to observe how the performances of V-Neighbor2Neighbor and UDVD are remarkably better than their respective single frame versions, namely Neighbor2Neighbor and Noise2Self, from where we can conclude that the usage of different successive frames leads to an improvement on the denoising performance.

7 Conclusions

At this point we can conclude that the best behaving model seems to be UDVD when it is trained using the MSE error, for it has both the lowest uMSE and highest SSIM. Nevertheless, V-Neighbor2Neighbor is also an approach to take into account, for it achieves a similar performance according to both metrics and has much faster training and inference times.

For that reason, a user friendly denoising code based on UDVD has been implemented to ensure that the physicists of the Crozier Research Group can run it correctly without needing any concrete knowledge on Deep Learning or Image Processing techniques.

This script has already been delivered to the group and one of their students is running it correctly in their servers, achieving satisfactory results to the date. After training the UDVD network on the whole dataset we have been able to provide them with a pretrained, general model so that they do not have to train the model from scratch every time but only tune it for some epochs.

In addition to the model training and denoiser implementation for the Crozier Research Group we have also achieved a very relevant result by developing and validating an unbiased estimator for the MSE metric in the absence of clean data and a well behaving SSIM approach.

More specifically we have developed and validated an unbiased MSE estimator, covering what was a gap of knowledge to the date, and also tested the performance of an unsupervised approach to SSIM, which does not have an accurate theoretical reasoning behind but has shown good results in the experiments performed.

This contribution will be part of a paper that will be presented to the NeurIPS Thirty-fifth Conference on Neural Information Processing Systems, together with the Crozier Research Group dataset, in the call for Datasets and Benchmarks of 2022.

On the other hand the uSSIM estimator failed the tests, most probably because of the treatment given to the SSIM formula in the uSSIM proof of Section 5.2.3, where instead of considering the expectancy of the estimator as a group only the expectancy of its elements was proven separately, which is clearly not a proof of unbiasedness.

The detection of this error is, nevertheless, another conclusion of the project, since part of its aim was the design and validation of new metrics and this one had a whole reasoning behind that has been reanalyzed before seeing the results.

In addition to all these academical results, a set of personal skills have also been developed or enhanced during the realization of this project.

8 Further steps

As this project was mainly self contained and most of the objectives have been fulfilled, there is not much more research remaining to be done in any of its aspects.

As new architectures and denoising techniques are published, some of their aspects could lead to new possibilities in the metrics design just as Neighbor2Neighbor inspired the best performing ones from the present work.

A valid, unbiased SSIM estimator remains to be designed by finding a different approach to its theoretical deduction after finding the error of the present one. To do so a deduction work has to be performed from the expected value of the classical SSIM formula in order to find the possible sources of bias and compensate them in the formula, a process that I personally do not know if is possible.

Finally, a V-Neighbor2Neighbor user friendly code remains to be implemented in order to offer the Crozier Research Group and the denoising community with a tool that levels with the present UDVD one in performance but provides a much higher computational efficiency.

Bibliography

Batson, J., & Royer, L. (2019). Noise2self: Blind denoising by self-supervision. In *International conference on machine learning* (pp. 524–533).

Benesty, J., Chen, J., & Huang, Y. (2010). Study of the widely linear wiener filter for noise reduction. In *2010 ieee international conference on acoustics, speech and signal processing* (p. 205-208).

Bergholm, F. (1987). Edge focusing. In *Ieee trans. putt. anal. mach. intellig., vol. pami-9* (p. 726-741).

Buendía, F., Gayoso-Cabada, J., & Sierra, J.-L. (2018). Using digital medical collections to support radiology training in e-learning platforms. In *European conference on technology enhanced learning* (pp. 566–569).

Deng, G., & Cahill, L. (1993, 01). An adaptive gaussian filter for noise reduction and edge detection. In (Vol. 3, p. 1615 - 1619 vol.3). doi: 10.1109/NSSMIC.1993.373563

Fleishman, S., Drori, I., & Cohen-Or, D. (2003, jul). Bilateral mesh denoising. *ACM Trans. Graph.*, *22*(3), 950–953. Retrieved from https://doi.org/10.1145/882262.882368 doi: 10.1145/882262.882368

Getreuer, P. (2013). A Survey of Gaussian Convolution Algorithms. *Image Processing On Line*, *3*, 286–310. (https://doi.org/10.5201/ipol.2013.87)

Gonzalez, R. C., Woods, R. E., & Masters, B. R. (2009). *Digital image processing*. Society of Photo-Optical Instrumentation Engineers.

Hou, X., Yuille, A., & Koch, C. (2013). Boundary detection benchmarking: Beyond f-measures. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 2123–2130).

Huang, T., Li, S., Jia, X., Lu, H., & Liu, J. (2021). Neighbor2neighbor: Self-supervised denoising from single noisy images. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition* (pp. 14781–14790).

Krull, A., Buchholz, T., & Jug, F. (2018). Noise2void - learning denoising from single noisy images. *CoRR*, *abs/1811.10980*. Retrieved from http://arxiv.org/abs/1811.10980

Lehtinen, J., Munkberg, J., Hasselgren, J., Laine, S., Karras, T., Aittala, M., & Aila, T. (2018). Noise2noise: Learning image restoration without clean data. *CoRR*, *abs/1803.04189*. Retrieved from http://arxiv.org/abs/1803.04189

Metzler, C. A., Mousavi, A., Heckel, R., & Baraniuk, R. G. (2020). *Unsupervised learning with stein's unbiased risk estimator.*

Mohan, S., Manzorro, R., Vincent, J. L., Tang, B., Sheth, D. Y., Simoncelli, E. P., … Fernandez-Granda, C. (2020). Deep denoising for scientific discovery: a case study in electron microscopy. *arXiv preprint arXiv:2010.12970*.

Mohan, S., Vincent, J., Manzorro, R., Crozier, P., Fernandez-Granda, C., & Simoncelli, E. (2021). Adaptive denoising via gaintuning. *Advances in Neural Information Processing Systems*, *34*.

Pitas, I., & Venetsanopoulos, A. N. (2013). *Nonlinear digital filters: principles and applications* (Vol. 84). Springer Science & Business Media.

Pont-Tuset, J., Perazzi, F., Caelles, S., Arbeláez, P., Sorkine-Hornung, A., & Van Gool, L. (2017). The 2017 davis challenge on video object segmentation. *arXiv preprint arXiv:1704.00675*.

Quan, Y., Chen, M., Pang, T., & Ji, H. (2020). Self2self with dropout: Learning self-supervised denoising from single image. In *2020 ieee/cvf conference on computer vision and pattern recognition (cvpr)* (p. 1887-1895). doi: 10.1109/CVPR42600.2020.00196

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, *323*(6088), 533–536.

Stein, C. M. (1981). Estimation of the Mean of a Multivariate Normal Distribution. *The Annals of Statistics*, *9*(6), 1135 – 1151. Retrieved from https://doi.org/10.1214/aos/1176345632

doi: 10.1214/aos/1176345632

Toderici, G., Vincent, D., Johnston, N., Jin Hwang, S., Minnen, D., Shor, J., & Covell, M. (2017). Full resolution image compression with recurrent neural networks. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 5306–5314).

Zhong, D., Yang, Y., & Du, X. (2018). Palmprint recognition using siamese network. In *Chinese conference on biometric recognition* (pp. 48–55).

Zuo, W., Zhang, K., & Zhang, L. (2018). Convolutional neural networks for image denoising and restoration. In M. Bertalmío (Ed.), *Denoising of photographic images and video: Fundamentals, open challenges and new trends* (pp. 93–123). Cham: Springer International Publishing. Retrieved from https://doi.org/10.1007/978-3-319-96029-6_4 doi: 10.1007/978-3-319-96029-6_4

9 Annex A

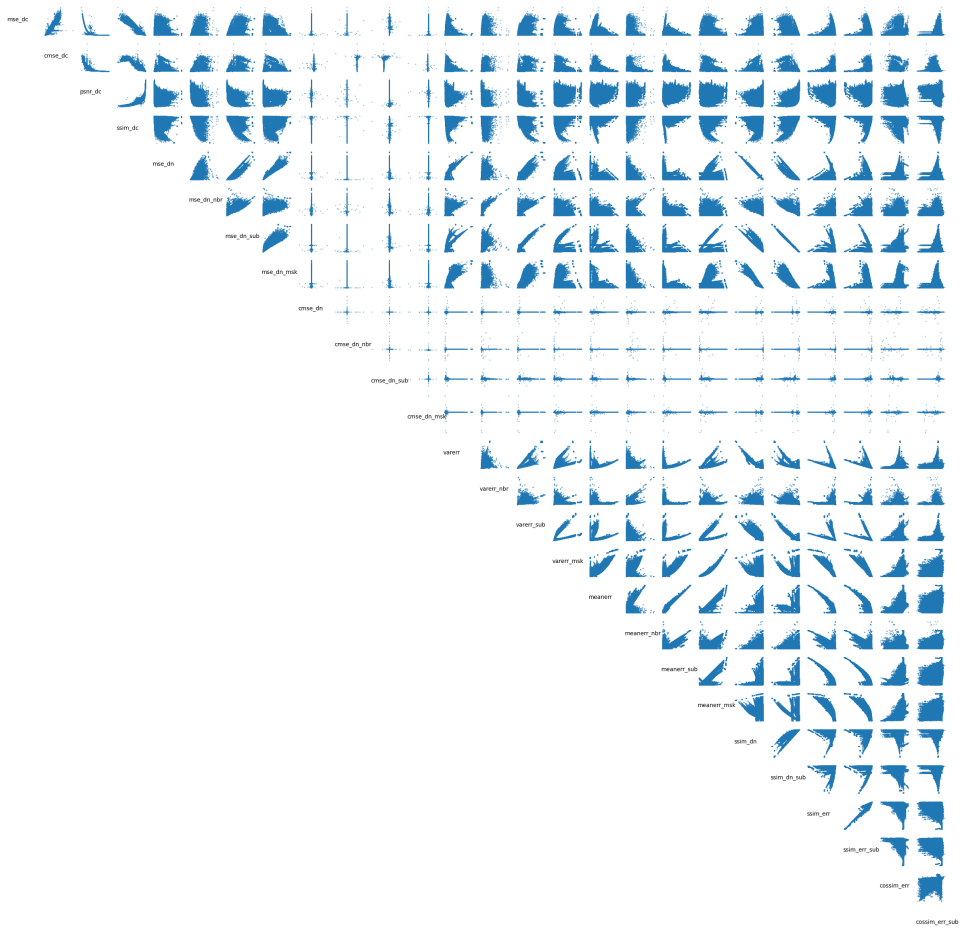

Figure 25: Scatterplot of all the metrics using all the images.

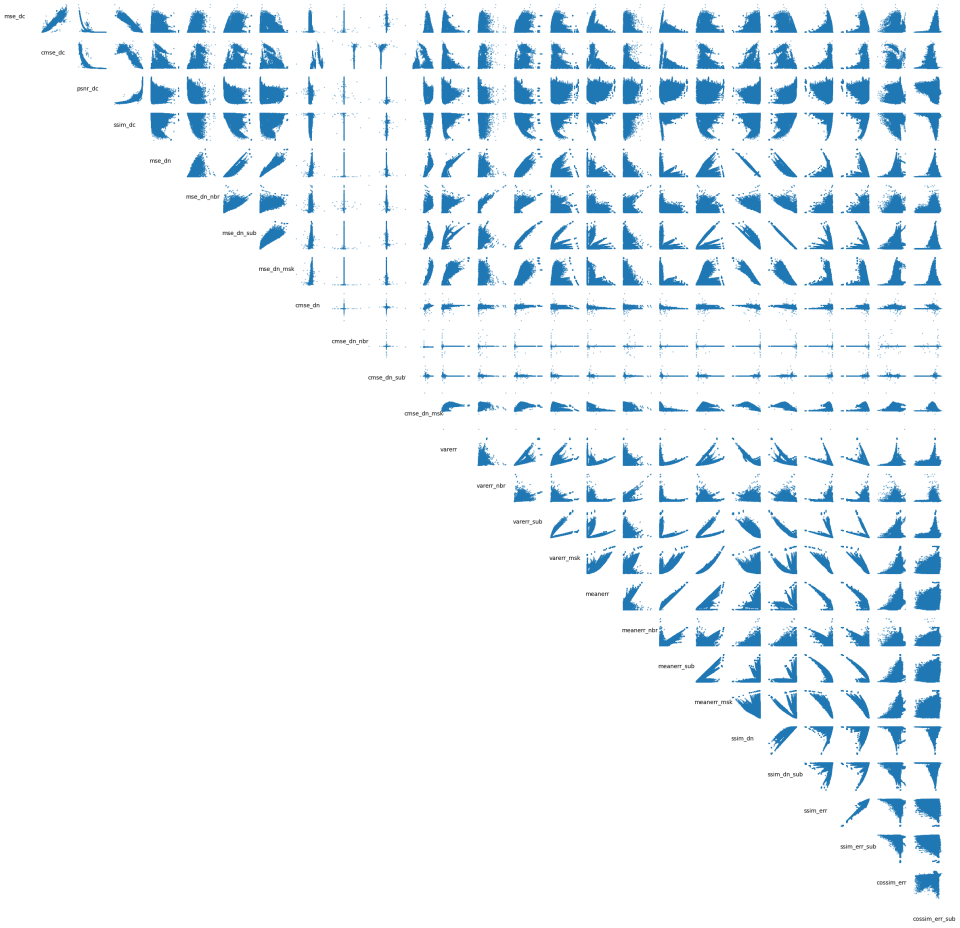

Figure 26: Scatterplot of all the metrics using only the images with Poisson and re-scaled Poisson noise.

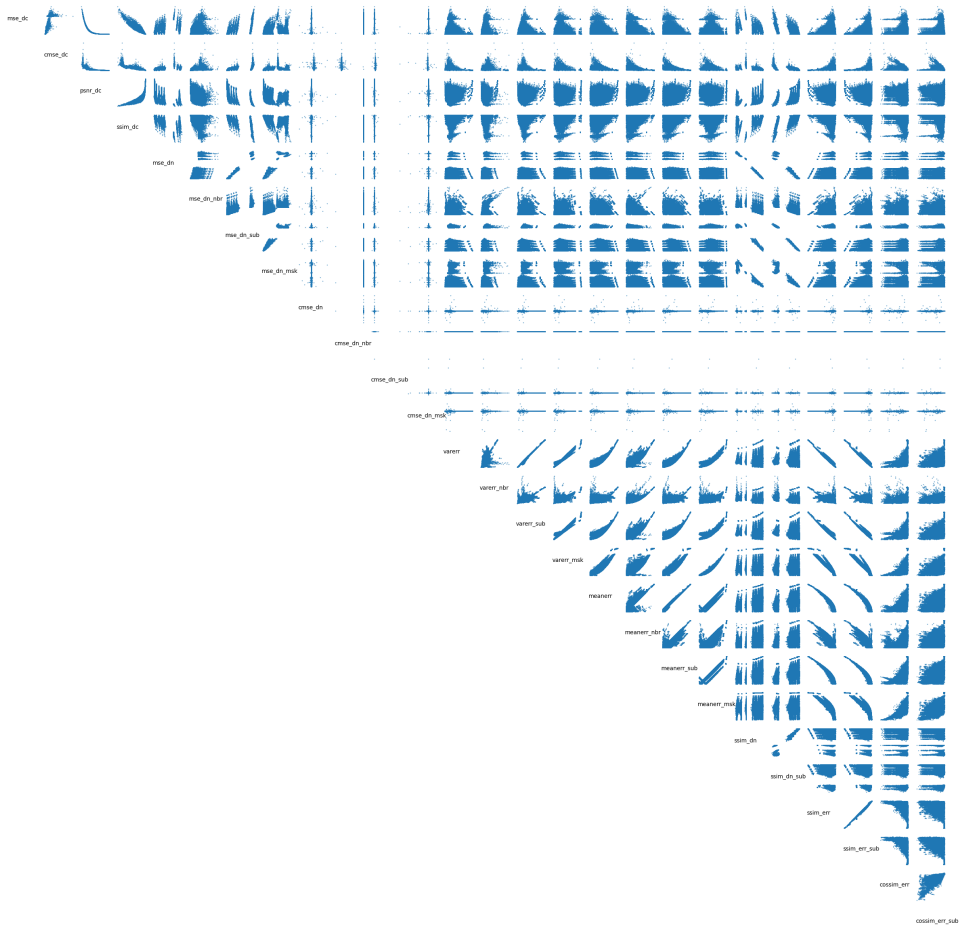

Figure 27: Scatterplot of all the metrics using only the images with Gaussian noise.

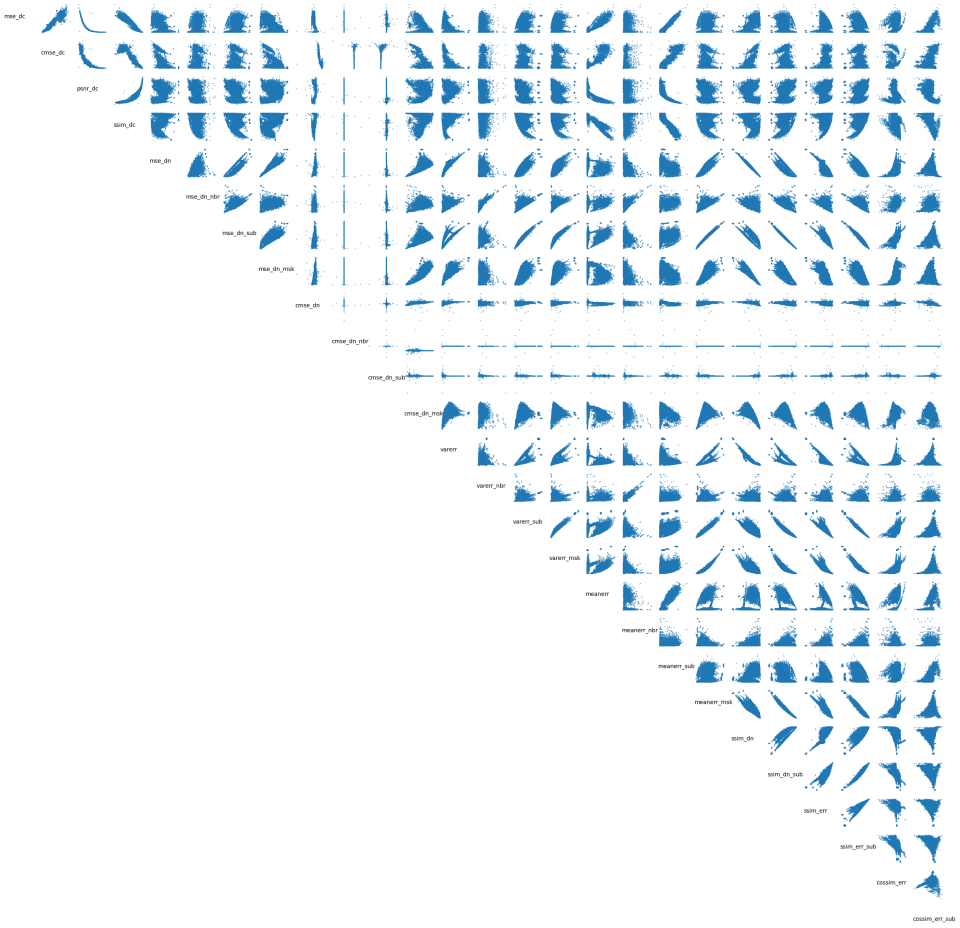

Figure 28: Scatterplot of all the metrics using only the images with pure Poisson noise.

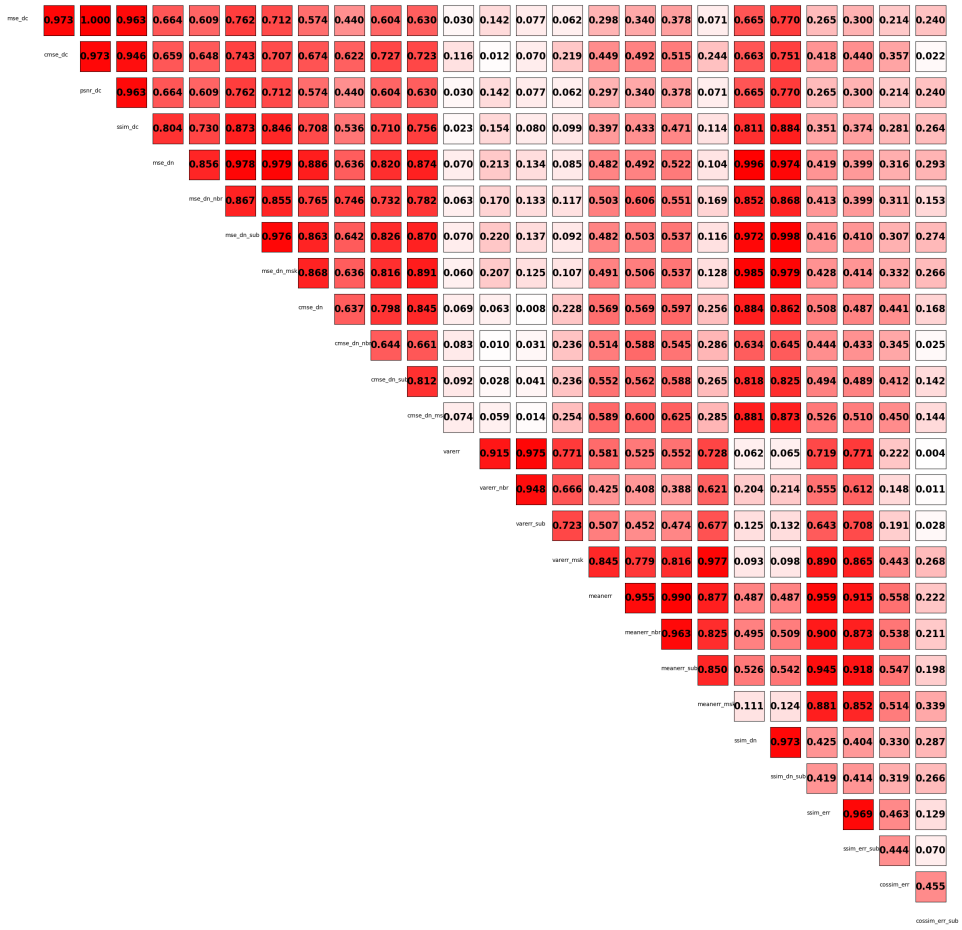

Figure 29: SRCC values of all the metrics using all the images.

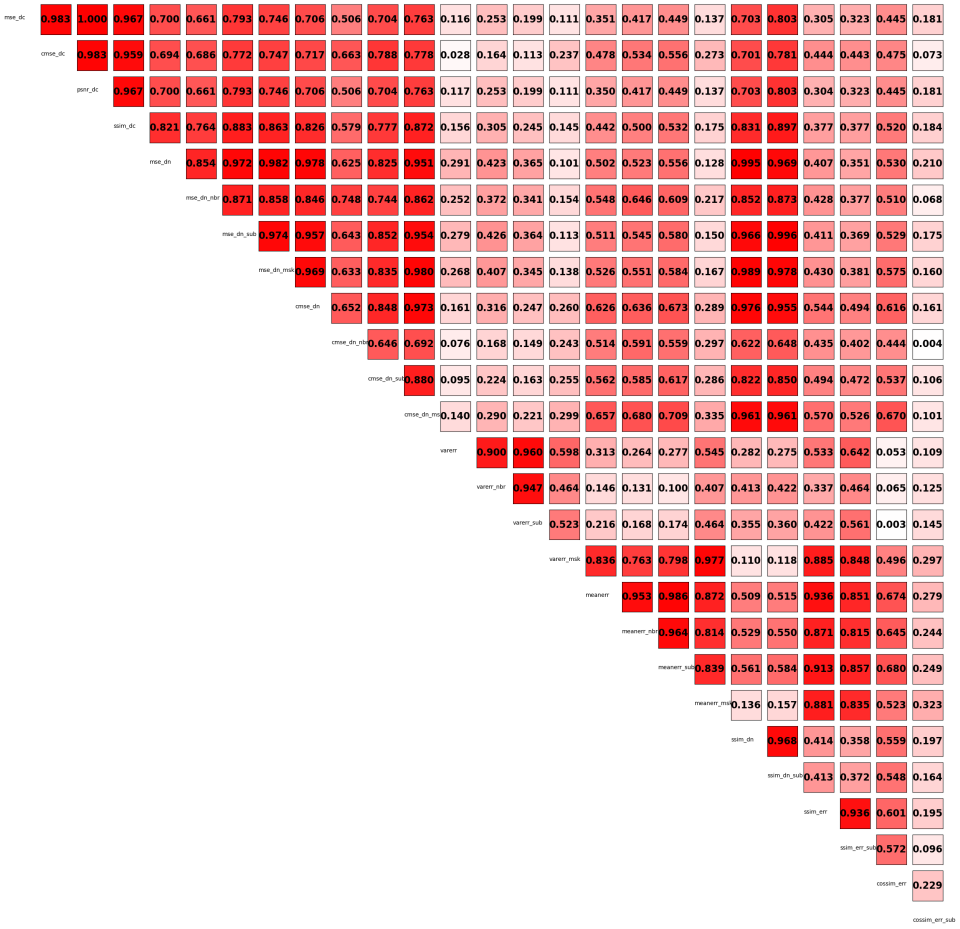

Figure 30: SRCC values of all the metrics using only the images with Poisson and re-scaled Poisson noise.

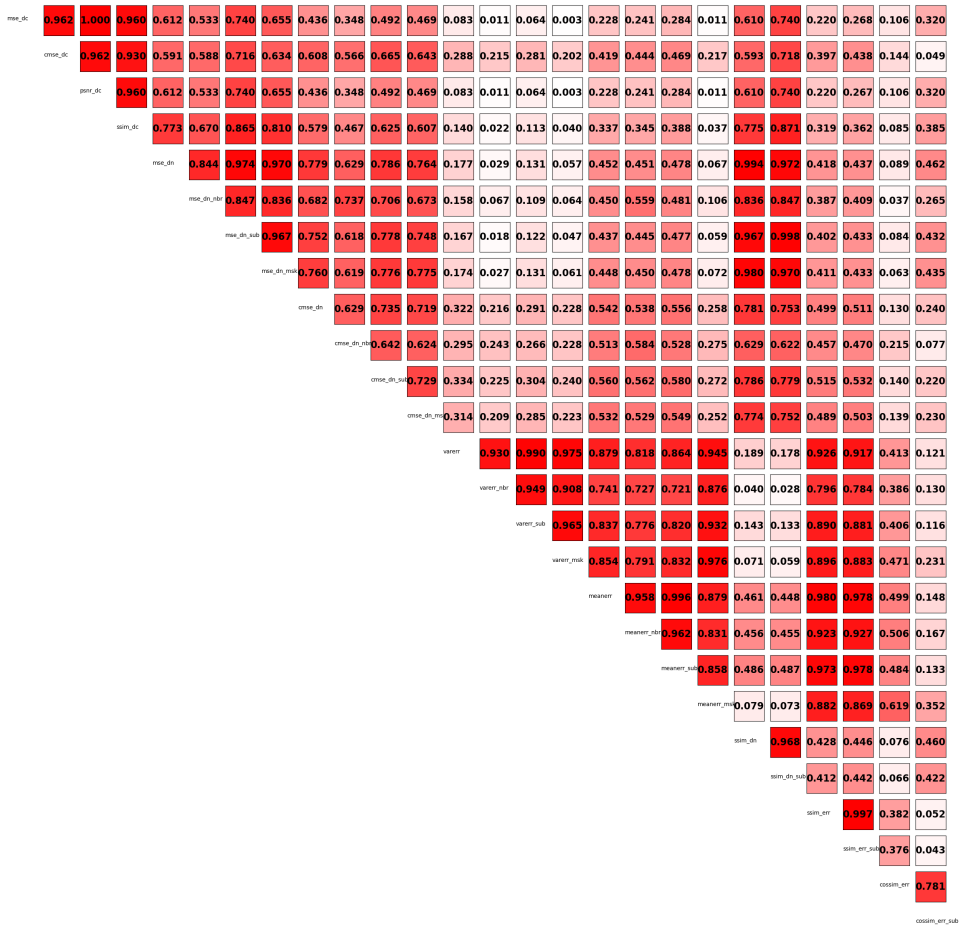

Figure 31: SRCC values of all the metrics using only the images with Gaussian noise.

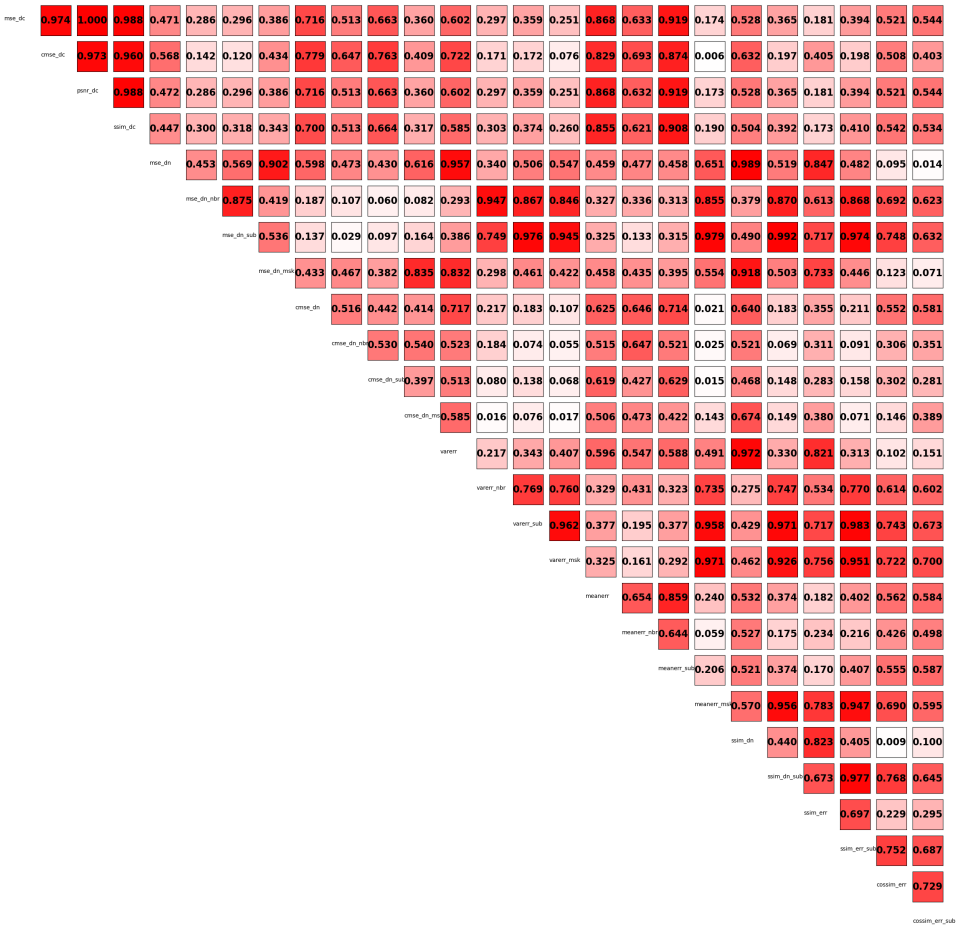

Figure 32: SRCC values of all the metrics using only the images with pure Poisson noise.

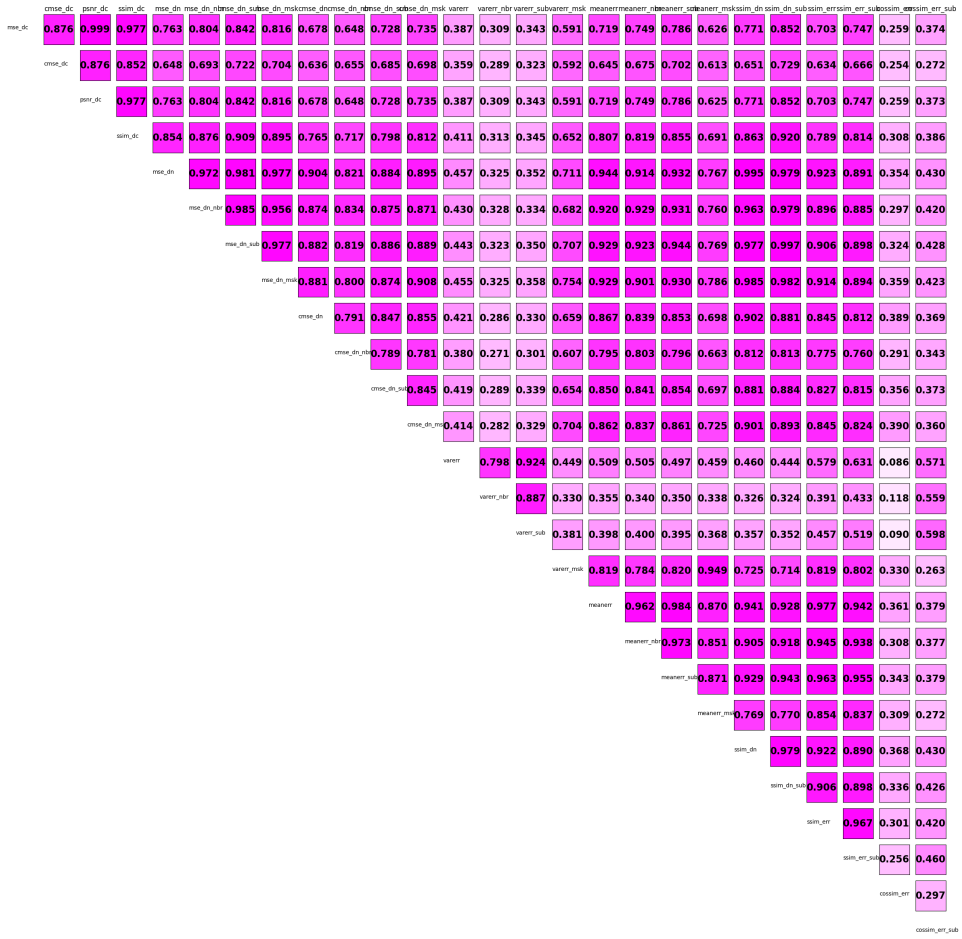

Figure 33: SRCCpI values of all the metrics using all the images.

Figure 34: SRCCpI values of all the metrics using only the images with Poisson and re-scaled Poisson noise.

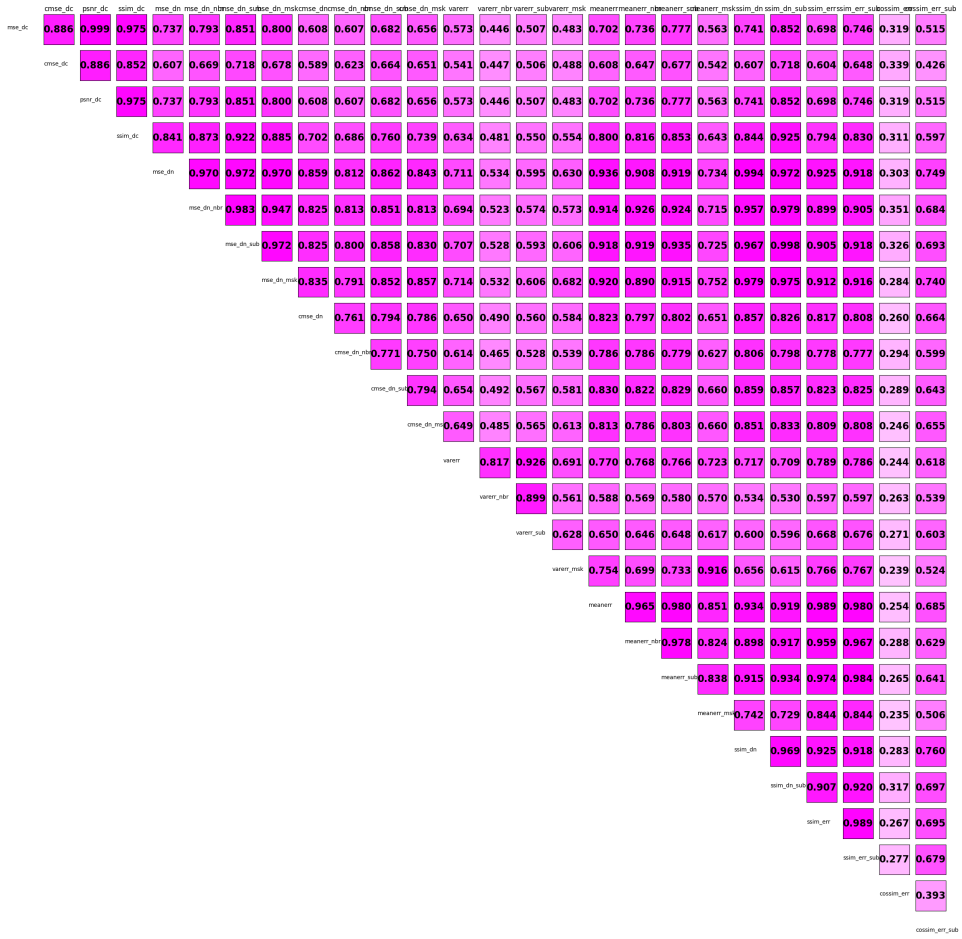

Figure 35: SRCCpI values of all the metrics using only the images with Gaussian noise.

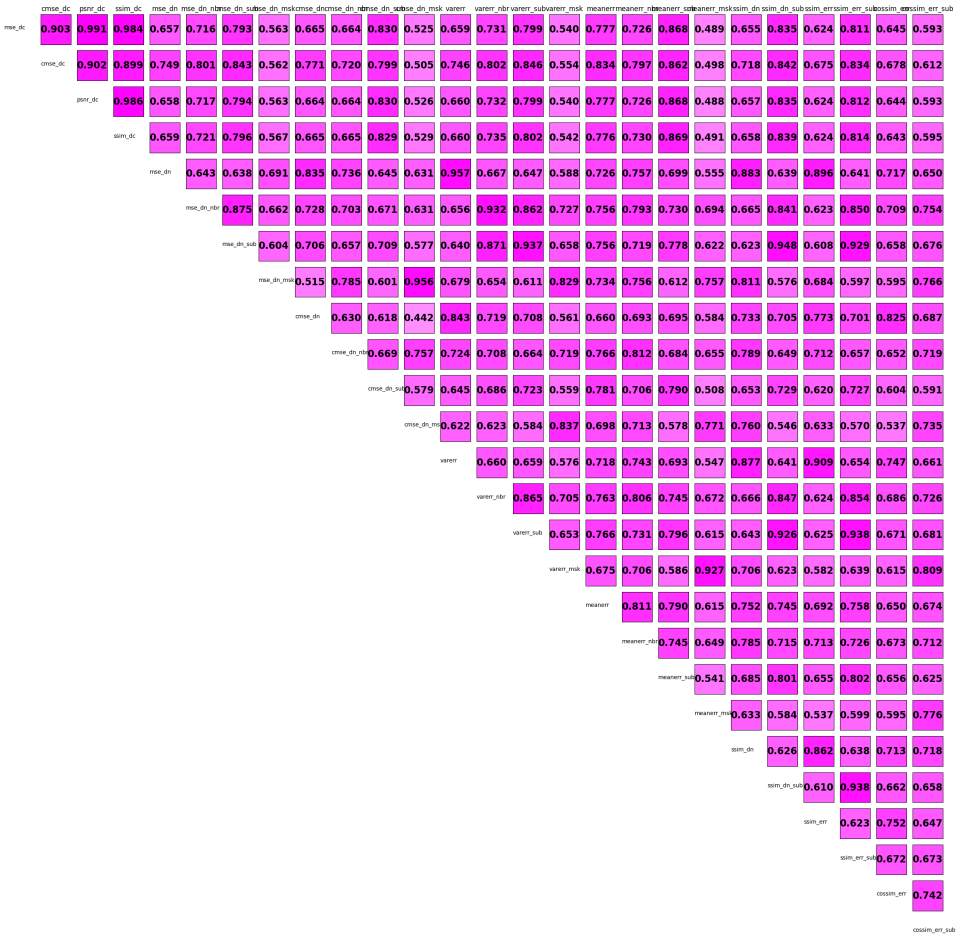

Figure 36: SRCCpI values of all the metrics using only the images with pure Poisson noise.

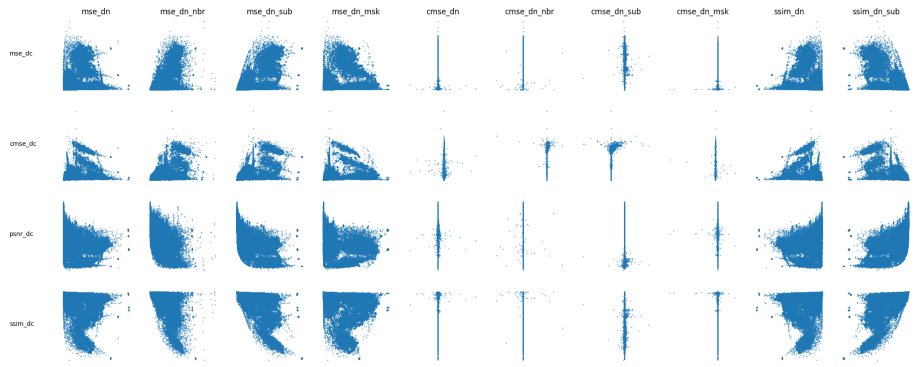

Figure 37: Scatterplot of the benchmark metrics using all the images.

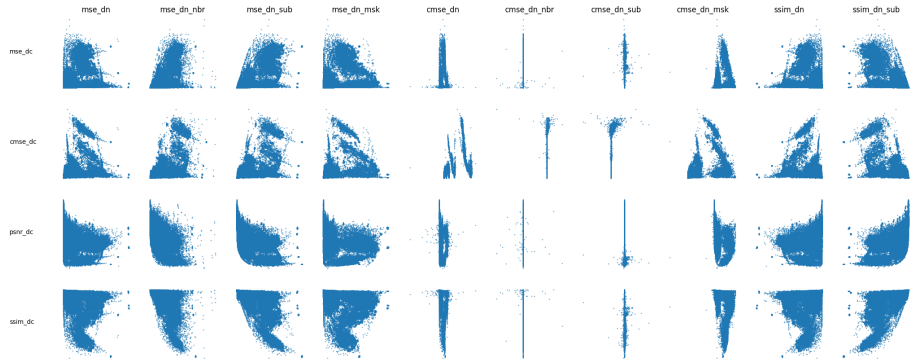

Figure 38: Scatterplot of the benchmark metrics using only the images with Poisson and re-scaled Poisson noise.

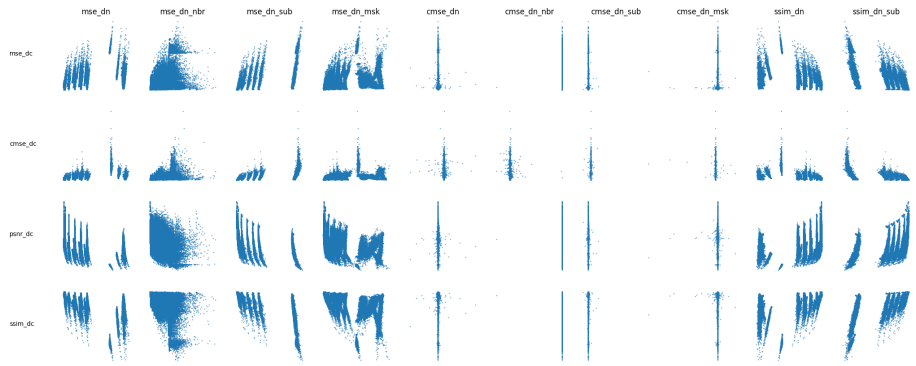

Figure 39: Scatterplot of the benchmark metrics using only the images with Gaussian noise.

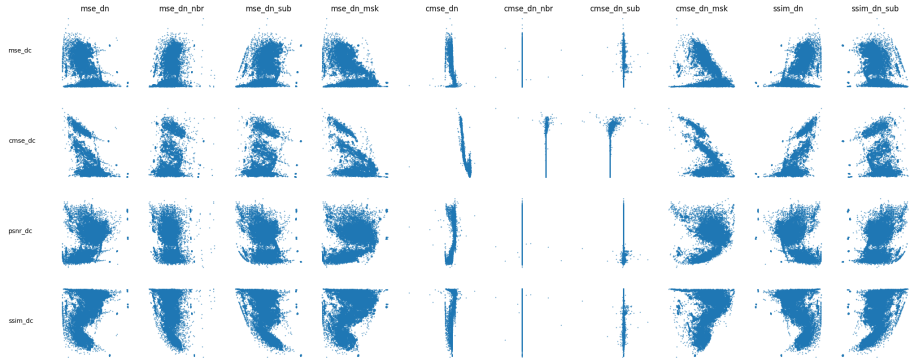

Figure 40: Scatterplot of the benchmark metrics using only the images with pure Poisson noise.

Figure 41: SRCC values of the benchmark metrics using all the images.

Figure 42: SRCC values of the benchmark metrics using only the images with Poisson and re-scaled Poisson noise.

Figure 43: SRCC values of the benchmark metrics using only the images with Gaussian noise.

Figure 44: SRCC values of the benchmark metrics using only the images with pure Poisson noise.

Figure 45: SRCCpI values of the benchmark metrics using all the images.

Figure 46: SRCCpI values of the benchmark metrics using only the images with Poisson and re-scaled Poisson noise.

Figure 47: SRCCpI values of the benchmark metrics using only the images with Gaussian noise.

Figure 48: SRCCpI values of the benchmark metrics using only the images with pure Poisson noise.

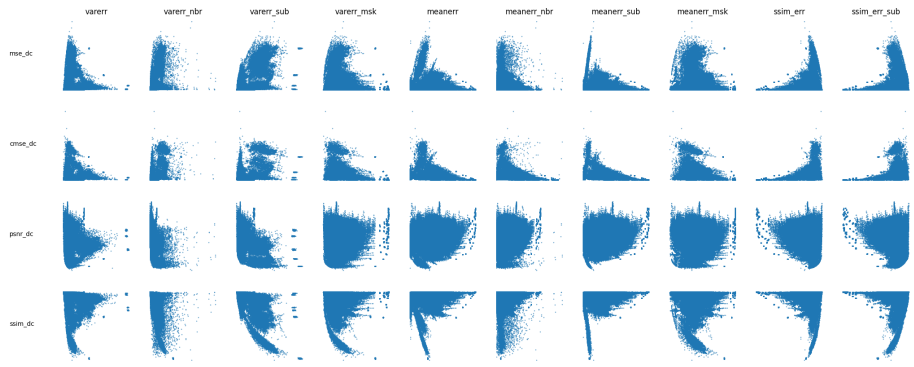

Figure 49: Scatterplot of the purposed metrics using all the images.

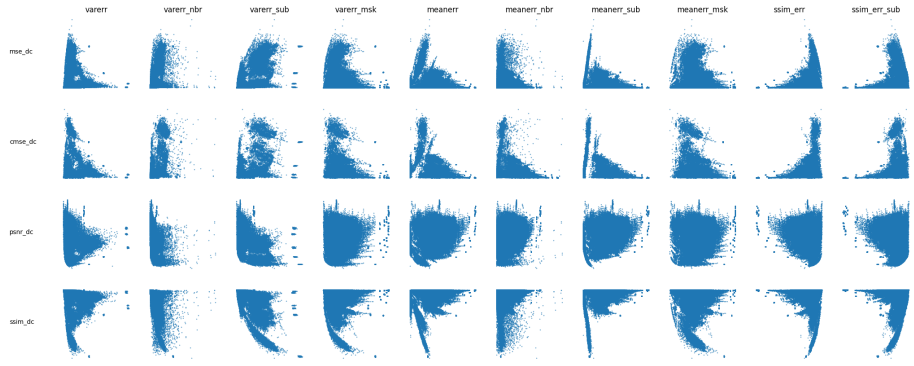

Figure 50: Scatterplot of the purposed metrics using only the images with Poisson and re-scaled Poisson noise.

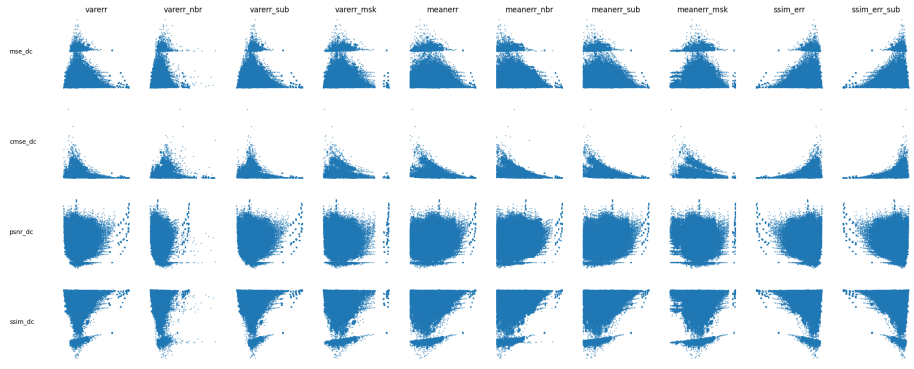

Figure 51: Scatterplot of the purposed metrics using only the images with Gaussian noise.

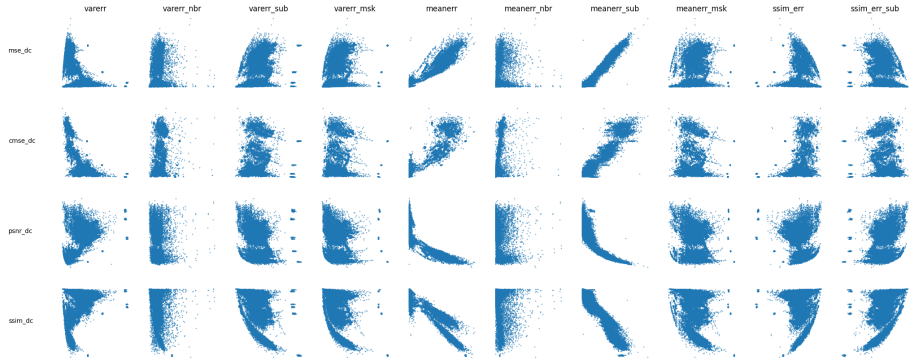

Figure 52: Scatterplot of the purposed metrics using only the images with pure Poisson noise.

	varerr	varerr_nbr	varerr_sub	varerr_msk	meanerr	meanerr_nbr	meanerr_sub	meanerr_msk	ssim_err	ssim_err_sub
mse_dc	0.0298	0.1415	0.0772	0.0618	0.2975	0.3403	0.3783	0.0713	0.2652	0.3001
cmse_dc	0.1162	0.0125	0.0703	0.2186	0.4493	0.4917	0.5155	0.2438	0.4185	0.4397
psnr_dc	0.0299	0.1416	0.0773	0.0617	0.2974	0.3402	0.3782	0.0712	0.2651	0.2999
ssim_dc	0.0232	0.1541	0.0804	0.0991	0.3967	0.4333	0.4706	0.1137	0.3511	0.3739

Figure 53: SRCC values of the purposed metrics using all the images.

	varerr	varerr_nbr	varerr_sub	varerr_msk	meanerr	meanerr_nbr	meanerr_sub	meanerr_msk	ssim_err	ssim_err_sub
mse_dc	0.1164	0.2533	0.1988	0.1109	0.3507	0.4169	0.4487	0.1371	0.3046	0.3234
cmse_dc	0.0279	0.1639	0.1135	0.2372	0.4777	0.5337	0.5557	0.2728	0.4440	0.4428
psnr_dc	0.1166	0.2534	0.1990	0.1107	0.3505	0.4168	0.4485	0.1369	0.3044	0.3232
ssim_dc	0.1561	0.3048	0.2454	0.1446	0.4417	0.4998	0.5321	0.1748	0.3774	0.3769

Figure 54: SRCC values of the purposed metrics using only the images with Poisson and re-scaled Poisson noise.

Figure 55: SRCC values of the purposed metrics using only the images with Gaussian noise.

Figure 56: SRCC values of the purposed metrics using only the images with pure Poisson noise.

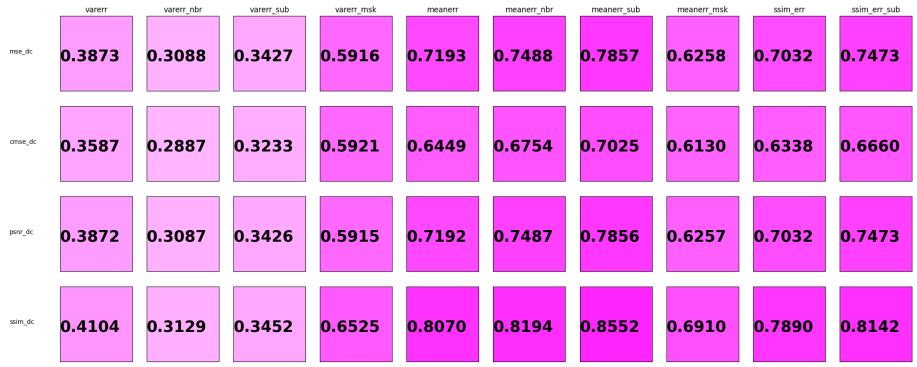

Figure 57: SRCCpI values of the purposed metrics using all the images.

Figure 58: SRCCpI values of the purposed metrics using only the images with Poisson and re-scaled Poisson noise.

Figure 59: SRCCpI values of the purposed metrics using only the images with Gaussian noise.

Figure 60: SRCCpI values of the purposed metrics using only the images with pure Poisson noise.