MASARYKOVA UNIVERZITA
Přírodovědecká fakulta
Ústav teoretické fyziky a astrofyziky

DIPLOMOVÁ PRÁCE

Brno 2020 Antónia Vojteková

MASARYKOVA UNIVERZITA

Přírodovědecká fakulta

Ústav teoretické fyziky a astrofyziky

REDUKCE ŠUMU ASTRONOMICKÝCH SNÍMKŮ ZA POMOCÍ NEURONOVÝCH SÍTÍ

DIPLOMOVÁ PRÁCE

ANTÓNIA VOJTEKOVÁ

VEDOUCÍ DIPLOMOVÉ PRÁCE:
Mgr. FILIP HROCH, Ph.D.

BRNO 2020

Bibliografický záznam

Autor:	Antónia Vojteková Přírodovědecká fakulta, Masarykova univerzita Ústav teoretické fyziky a astrofyziky

Název práce:	Redukce šumu astronomických snímků za pomocí neuronových sítí
Studijní program:	Teoretická fyzika a astrofyzika
Obor:	Astrofyzika
Vedoucí práce:	Mgr. Filip Hroch, Ph.D.
Akademický rok:	2019/2020
Počet stran:	ix + 90
Klíčová slova:	průzkum, unet, šum obrázků, neuronová síť

Bibliographic record

Author:	Antónia Vojteková Faculty of Science, Masaryk University Department of Theoretical Physics and Astrophysics

Title of Thesis:	Neural network noise reduction of astronomical images
Degree Programme:	Theoretical physics and astrophysics
Field of study:	Astrophysics
Supervisor:	Mgr. Filip Hroch, Ph.D.
Academic Year:	2019/2020
Number of Pages:	ix + 90
Keywords:	survey, image noise, unet, neural network

ABSTRAKT

Astronomické snímky jsou klíčové pro zkoumání, a porozumění našemu Vesmíru. Získání jejich dostatečného počtu je časově i přístrojově velmi náročné. Mohutné optické přístroje, například Hubblův kosmický dalekohled, přitahují velké množství astronomů. Počty projektů tak až šestkrát převyšují kapacity tohoto přístroje. Proto hledáme způsoby, jak využít stávající data.

Snímky často obsahují dodatečný šum, který přikládá důležitosti dodatečnému zpracování a analýze dat. To vede k použití metod strojového učení ke získání maximálního využití dat ze snímků. Uvažujeme dvě metody známé jako Astro U-net, a také fully-convolution neural networks, k redukci šumu a zlepšení vzhledu snímků. Předkládaný přístup odpovídá expozicím s dvojnásobným časem, minimálním vychýlením a ztrátou informace.

ABSTRACT

Astronomical images are essential for exploring and understanding the universe. To obtain the needed amount of images is time-consuming and expensive. Deep optical telescopes such Hubble Space Telescope, attracts vast numbers of astronomers that want to use it, however, the current oversubscription rate is six-to-one. This means that the amount of observation time requested is six times larger than the time awarded on the telescope.

Images also often contain additive noise, which makes mandatory denoising step in post-processing data before further data analysis. To maximise the efficiency and information gain in the post-processing of astronomical imaging, we decided to turn on machine learning. We propose Astro U-net, a fully-convolution neural network for image denoising and enhancement. Our approach can produce images with double observation time and with minimum bias or information loss.

MUNI
SCI

MASARYKOVA UNIVERZITA
PŘÍRODOVĚDECKÁ FAKULTA
KOTLÁŘSKÁ 2, 611 37 BRNO
IČ: 00216224
DIČ: CZ00216224

ZADÁNÍ
DIPLOMOVÉ PRÁCE

Akademický rok: 2019/2020

Ústav:	Ústav teoretické fyziky a astrofyziky
Studentka:	Bc. Antónia Vojteková
Program:	Fyzika
Obor:	Teoretická fyzika a astrofyzika
Směr:	Astrofyzika

Ředitel *Ústavu teoretické fyziky a astrofyziky* PřF MU Vám ve smyslu Studijního a zkušebního řádu MU určuje diplomovou práci s názvem:

Název práce:	Redukce šumu astronomických snímků za pomocí neuronových sítí
Název práce anglicky:	Neuron network noise reduction of astronomical images
Jazyk závěrečné práce:	

Oficiální zadání:
Všudypřítomný fotonový šum je jednou se základních vlastností astronomických snímků. Od něj se odvíjí konstrukce astronomických přístrojů, návrh pozorovacích programů, a také konečné zpracování snímků. Astronomové proto po léta hledají způsoby, jak, za pomocí rafinovaných statistických metod, získat ze snímků maximální množství informací.

Jeden z moderně se rozvíjejících směrů je aplikace Bayesovského statistického přístupu v podobně tzv. neuronových sítí.

Cílem práce je aplikace vlastnoručně sestrojeného algoritmu na vybraná astronomická data. K tomu bude třeba využít nejnovějších poznatků matematických, fyzikálních a počítačových metod, především pak metod strojového učení.

Vedoucí práce:	Mgr. Filip Hroch, Ph.D.
Datum zadání práce:	19. 5. 2020
V Brně dne:	27. 5. 2020

Poďakovanie

I'm a greater believer in luck,
and I find the harder I work the
more I have of it.

Thomas Jefferson

I want to say thank you to my supervisors Filip, Maggie, Ivan, Lyndsay and Qifeng. Also, I would like to thank ESAC and everybody whom I met there for a beautiful moment while working on this project. I'm grateful also for my university, which helped me to grow not just in my career-wise, but also as a person.
My family and friends also deserve thanks, especially my beloved partner, Andreas.
For GPU power I would like to thank ESA, ESTEC, project Sandy and to my university and for data the Hubble space telescope.

Čestné prohlášení

Prohlašuji, že jsem svoji diplomovou práci vypracovala samostatně pod vedením vedoucího práce s využitím informačních zdrojů, které jsou v práci citovány.

Brno 25. května 2020

Podpis autora

Contents

1 Introduction **1**

2 Method **2**
2.1 Neural network . . . 2
2.2 Convolutional Neural network . . . 3
2.3 How to create the architecture . . . 5
2.3.1 Size of network and data-set matters . . . 5
2.3.2 Activation function . . . 8
2.3.3 Loss function . . . 9
2.3.4 Back-propagation . . . 9
2.4 Architecture . . . 11
2.5 Training process . . . 11
2.6 Evaluation metrics . . . 13
2.6.1 Common metrics . . . 13
2.6.2 The source detection . . . 14
2.6.3 The intensity distribution . . . 15

3 Hubble space telescope data **18**
3.1 Training data . . . 20
3.1.1 Simulated data . . . 20
3.1.2 Output data . . . 24

4 Results **26**
4.1 Experiments . . . 26
4.1.1 Loss function . . . 26
4.1.2 Number of input channels . . . 26
4.1.3 Segmentation map . . . 27
4.1.4 Exposure time ratio . . . 27
4.1.5 Activation function . . . 28
4.2 Network Evaluation . . . 30
4.2.1 Source detection . . . 31
4.2.2 Randomness in the flux recovery . . . 43
4.2.3 The intensity Distribution . . . 51
4.2.4 Non-Astronomical image input . . . 72
4.2.5 Real data . . . 76

4.3 What is inside matters 80

5 Conclusion and future work **86**

Bibliography **88**

Introduction

In every field of science, data are the crucial element to make a breakthrough and to prove the theory. The history of astronomy, we see massive progress, how we obtain the data and how we process it. An observation of the position of the stars and planets helped our ancestors to predict the weather, even when they did not fully understand how and why it works. Once upon the time time, they were able to make a more sophisticated prediction – these stars are visible in a specific time of year because the Earth orbits the Sun and in that part of a year we can expect floods.

Edwin Hubble discovered new galaxies by poring over photographic plates, but how to identify all galaxies in Hubble Space Telescope archive? Nowadays, developing state-of-art telescope and cameras is very challenging. With new technologies, we faced new challenges. The data is no use if we cannot turn them into knowledge. That is the reason why humankind seeks for help, and answers, in the new scientific field – machine learning. Machine learning can help us to sort a big number of data, classify objects on the images (galaxies, nebulaes), help us with computationally heavy simulations [Rodríguez et al., 2018, Schawinski et al., 2017, Zingales and Waldmann, 2018], and maybe one day it can help us solved the most important question about our existence. We can just hope that answer will be not *42* [Adams, 1995].

Currently, we obtain high-quality data from long observations or from stacking multiple images with shorter exposure time, to increase the depth and to reduce noise in the image. This approach helps to increase the signal-to-noise ratio (SNR); however, it requires long observation and extended post-processing to select, align and combine the images [Zackay and Ofek, 2017a,b]. In our work, we turn to the machine learning in belief that one can help with reducing observation time by denoising and enhancing the images.

Method

2.1 Neural network

In 1943 the first formal model of a neuron was proposed by McCulloch and Walter Pitts. Their neuron worked like a logic gate. The neuron acted as OR, AND, NOT logical operations; thus enable the network to act like a computer. The model lack essential feature – learnable weight.

In 1950s Frank Rosenblatt introduced a perceptron. In this model, neurons were connected by variable weights: the output is one of the weighted sums of its input is above threshold, and zero if it is below. The first implementation of the perceptron was not by software, because it was prolonged back then. Rosenblatt decided to build his device were weights were implemented by variable resistors, and learning the weights was done by electric motors that turned the knobs on the resistor [Domingos, 2018].

The fundamental principle of the neural network is usually compared with the basic block of our brain – neuron. The basic unit of the neural network is called a neuron. Every neuron n is connected trough weighted connection w_n (Figure 2.1). Input x_n is multiplied by weight w_n and sum up

$$z = \sum_n w_n x_n + b_n, \tag{2.1}$$

where b_n is a bias which helps to improve the performance of the network. Because in neural networks, one is usually dealing with a large number of inputs, and weights, it is convenient to write Equation 2.1 in matrix multiplication form, where bias is inserted into weight ($\mathbf{W}$), and input ($\mathbf{X}$) matrix on the zero position

$$z = \begin{bmatrix} \dots & w_0^T & \dots \\ \dots & w_1^T & \dots \\ \dots & w_2^T & \dots \\ & \vdots & \\ \dots & w_m^T & \dots \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}$$

or

$$z = \mathbf{W}^T \mathbf{X}. \tag{2.2}$$

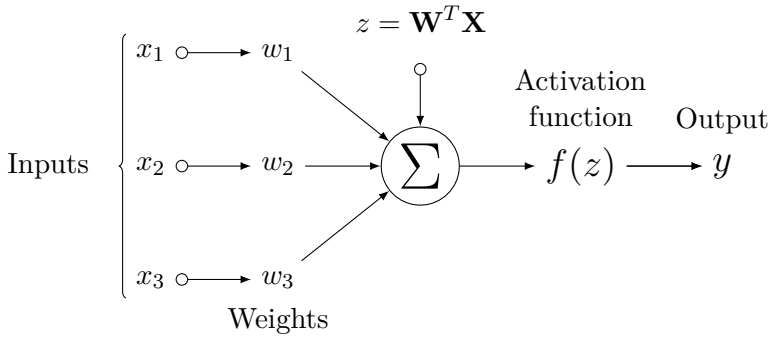

Figure 2.1: Scheme of the neuron

Next step is application of an activation function on output $y = f(z)$. The activation function is an important part of the neural network; without it the neural network would work like a linear transformation of its input. The function introduces non-linearity to the network. If we want to create the neural network we need to put basic units, neurons, in hierarchy structures – layers. The first layer is called the input layer, it takes our training data. The following layers are hidden layers. The number of hidden layers, and neurons in it, are optional. We can choose how big network we want to create depending on our problem (Subsubsection 2.3.1).

The input layer contains input data. The input information goes to the first hidden layer where matrix multiplications are done; then the output is used as an input for the second hidden layer et cetera. The output layer, the last layer of the network, contains the output information, e.g. our input belongs to class A or class B.

In the beginning, the neural network weights are initialised randomly to non-zeros small values. This values must be different; otherwise, training of the network would fail. During the training of the network, the information goes from the input layer through hidden layers to the output layer. After the last layer, a *loss function* is used to evaluate the performance of the network. The loss function calculates an error of the output. The last step is the *backpropagation*; errors are propagated to the network, and weights are change [Goodfellow et al., 2016, Raschka, 2015].

2.2 Convolutional Neural network

Convolutional neural network (CNN) is suitable to process image-like data, which can be represented as a matrix with three dimensions – height × width

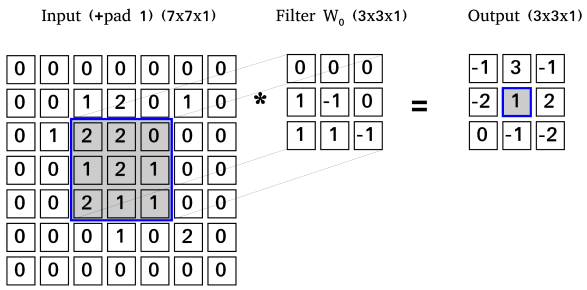

Figure 2.2: Example of convolution. Filter slide on the input with stride two and in every position filter is multiplied with the local regions of the input and then it sums up and produce output.

$\times$ depth. These type of networks use a special mathematics operation called convolution:

$$z(t) = x(t) * y(t) = \int_{-\infty}^{\infty} x(\tau)y(t-\tau)d\tau. \tag{2.3}$$

Like neural network (Subsection 2.1), CNN has the input – *input feature maps*, weights – *sets of filters* and output – *output feature maps*, where all of the mentioned are matrices with three dimensions. Sets of filters contains learnable parameters, which are changed while training. Applying the convolution input feature map are convoluted with set of filters and result is output feature map (Figure 2.2):

$$O = (I * F)(i,j) = \sum_m \sum_n I_{m,n} F_{i-m,j-n}. \tag{2.4}$$

An amount of output feature maps is influenced by a number of filters, it is the same as the number of filters. Both height and width are a result of spacial dimension of filters, stride and zero-padding. Zero-padding (P) is boarder around input which contains just zeroes. Stride (S) is a step with which is filter moved over the input map. Width (W) or height (H) of output can be calculated as:

$$W_{\text{output}} = \frac{W_{\text{input}} - F_{\text{W}} + 2P}{S} + 1. \tag{2.5}$$

Convolution layer can reduce spacial size of the input but more often a pooling layer in employ. Pooling is applied on every feature map separately,

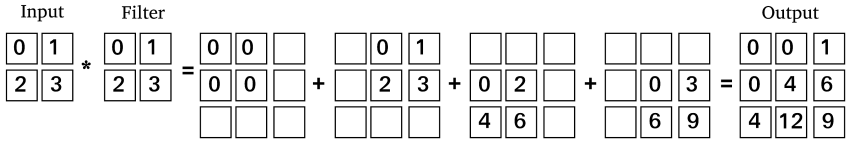

Figure 2.3: Example of transpose convolution operation. This diagram shows one way how to explain this operation, another explanation is Figure 2.4

preserving the important features in the map. Also, it helps to reduce computational time.

Another type of layer is a transpose convolution [Dumoulin and Visin, 2016, Odena et al., 2016] sometimes incorrectly named as a deconvolution. The goal of the layer is to increase the spacial size of input layer – it work as up-sample layer (Figure 2.3). Wildly known types of up-sample layers are nearest neighbour and bilinear up-sampling. These algorithms used common interpolations techniques, which estimates values of newly added pixels. Advantage of the transpose convolution are learnable filters which estimate new pixels. Convolution and transpose convolution can be also represented as matrix multiplication. As we can see at Figure 2.4, the convolution is matrix multiplication of filter and the input and in transpose convolution (Figure 2.5), we multiply a *transpose* filter and the input.

In the case of image-to-image translation networks [Chen et al., 2017, Zhang et al., 2019] are usually fully-convolutional neural network [Long et al., 2014] containing convolutional, transpose convolutional (up-sampling) and pooling layer [Goodfellow et al., 2016].

2.3 How to create the architecture

There is no rule on how to create suitable architecture. Usually, more types of architecture are created, and then their performances are evaluated. Sometimes the architecture used for a similar problem can give us a hint on how to move forward.

2.3.1 Size of network and data-set matters

One of the common problems is overfitting of the neural network. Overfitting means that the network has good results on training data but bad results on validation data (data unseen during training). The opposite problem

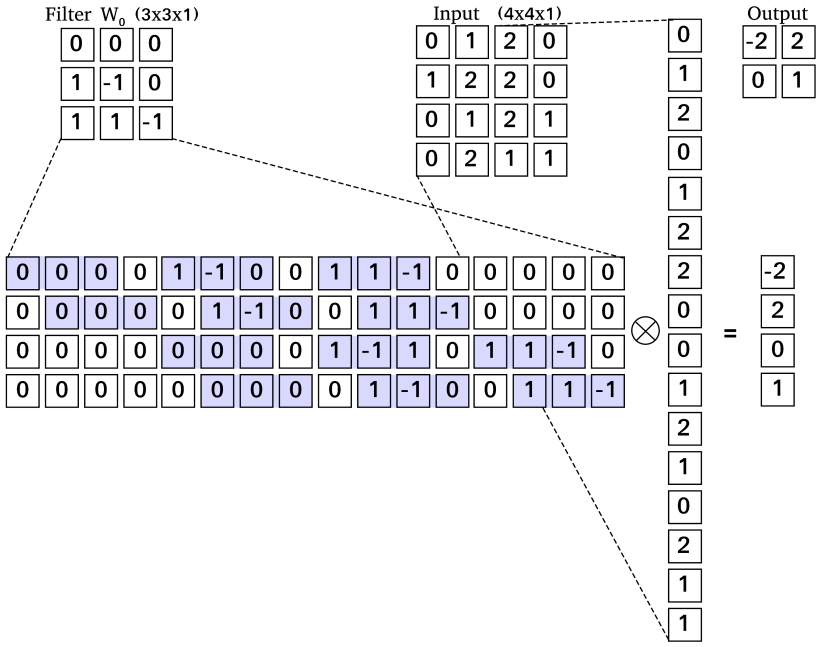

Figure 2.4: Example of the convolution operation represented as matrix multiplication. The filter is transformed into the matrix, the input into the vector and after multiplication, we can transform the output back into the form of the output feature map.

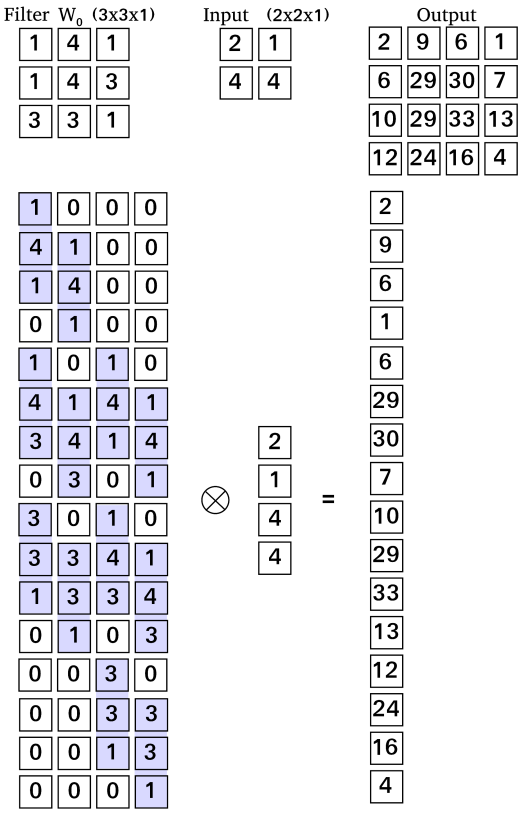

Figure 2.5: Example of the transpose convolution, similar as in Figure 2.4 operation is represented as matrix multiplication but with transposed filter matrix – from there name transpose convolution is derived.

is underfitting; the network is not complex enough to capture pattern in training data and performance is low on both training and validation data.

These problems can be caused by the size of the neural network and the data-set. Good data-set is fundamental for the training of the neural network. With small data-set, the big network could be able to memorise it and cause overfitting. On the other hand, a small network would lack the ability to fit the data then model would suffer from underfitting. The first step before training of the network is to answer questions:

- *Is data-set suitable for training?*
- *Is the architecture of the network suitable for data-set?*

2.3.2 Activation function

Activation functions bring non-linearity into the neural network. The function has to be differentiable to learn the weights that connect the neurons using a gradient-based approach. There are many different types of the activation function – sigmoid, hyperbolic tangent, ReLU [Nair and Hinton, 2010], LeakyReLU, Parametric ReLU (PReLU) [He et al., 2015], self-gated activation function (SWISH) [Ramachandran et al., 2017], etc...

In our project, we experimented with four activation functions. Leaky Rectified Linear Unit (LeakyReLU) is one of the most wildly used activation function. The advantages over other previous proposed activation function is that it is an easily optimized, monotonic function, unbounded above and bounded below. LeakyReLU was designed to avoid zero gradients, it leaks small negative numbers. Functions are defined as:

$$f(x) = \begin{cases} x, & x > 0 \\ ax, & x \leq 0 \end{cases} \tag{2.6}$$

where a is zero for ReLU and small constant number for LeakyReLU. The difference between LeakyReLU and PReLU is that for PReLU a is not constant but a trainable parameter. SWISH is non-monotonic function, which is same as LeakyReLU unbounded above and bounded below. It is defined as:

$$f(x) = x \cdot \text{sigmoid}(\beta \cdot x), \tag{2.7}$$

where $\beta > 0$ can be trainable or constant. In our work, it is set to constant equals to one.

2.3.3 Loss function

In case of supervised learning, the data-set has three parts – the input, the output and the ground truth. To evaluate performance of the network we must to define a loss function. One compares the output of the network with the ground truth. In our work we make experiments with three loss functions – $L1$, $L2$ and perceptual loss [Johnson et al., 2016]:

$$L_1 = \frac{1}{N}|y - y'|, \tag{2.8}$$

$$L_2 = \frac{1}{N}(y - y')^2, \tag{2.9}$$

where N denotes the number of pixels and y, y' the output image and ground truth respectively. To calculate perceptual loss, one has to have a pre-trained neural network (VGG-19). The output of the network is compared not just with ground truth, but also output and ground truth serves as the input of the pre-trained network, and there selected feature maps are compared. This loss can measure high-level perceptual and semantic differences between images.

2.3.4 Back-propagation

In the beginning, all weights in the neural network are initialised randomly. To adapt it to our problem we have to train it. During training, information flows through the network and in the end, the loss function is used to evaluate the performance of the network. It is important to pick loss function suitable to the problem. Zhao et al. [2017] discuss the most common loss functions L_1 and L_2. After the loss calculation the optimising algorithm is used to change weights in the network. The basic one is called gradient descent – it takes the opposite direction of the gradient of the loss function with respect to weights

$$\mathbf{w}^t = \mathbf{w}^{t-1} - \alpha \nabla L(\mathbf{w}), \tag{2.10}$$

where $\mathbf{w}$ denotes weight matrix in current t and previous $t-1$ epoch and L is the loss function and α is a learning rate. The learning rate is one of the hyper-parameters, which we have to choose before training of the network. To demonstrate how the backpropagation works, we have neural network with input as in Figure 2.6. If we want to update w_5 we have to calculate:

$$\frac{\partial L}{\partial w_5} = \frac{\partial L}{\partial y'}\frac{\partial y'}{\partial z}\frac{\partial z}{\partial w_5} \tag{2.11}$$

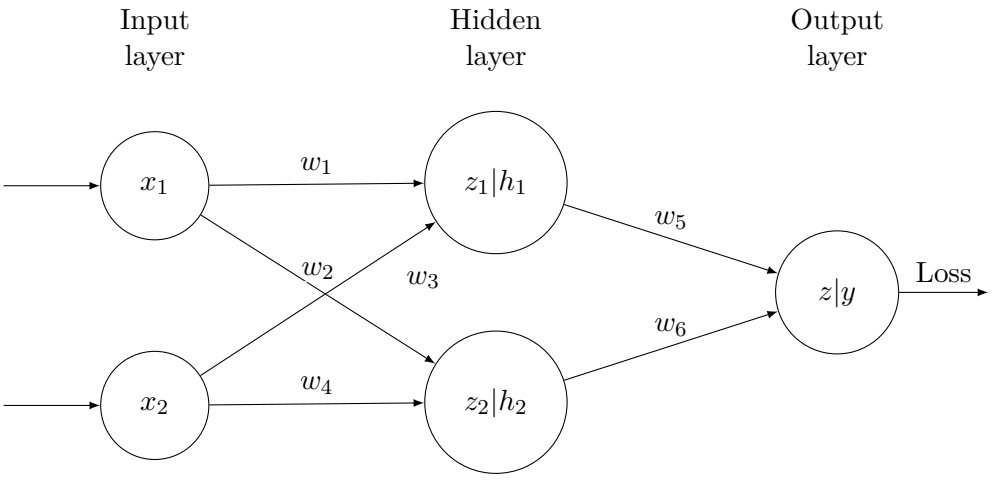

Figure 2.6: Example of the neural network with one hidden layer.

$$w_5^t = w_5^{t-1} - \alpha \frac{\partial L}{\partial y'} \frac{\partial y'}{\partial z} \frac{\partial z}{\partial w_5}. \tag{2.12}$$

And in same way to update w_1 weight:

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial y'} \frac{\partial y'}{\partial z} \frac{\partial z}{\partial h_1} \frac{\partial h_1}{\partial z_1} \frac{\partial z_1}{\partial w_1}, \tag{2.13}$$

$$w_1^t = w_1^{t-1} - \alpha \frac{\partial L}{\partial y'} \frac{\partial y'}{\partial z} \frac{\partial z}{\partial h_1} \frac{\partial h_1}{\partial z_1} \frac{\partial z_1}{\partial w_1} \tag{2.14}$$

where h_1 and z_1 defined as:

$$h_1 = f_A(z_1), \tag{2.15}$$

$$z_1 = x_1 \cdot w_1 + x_2 \cdot w_3, \tag{2.16}$$

f_A is an activation function. During years, many different optimising algorithms were proposed and many of them helped to speed-up the training and obtain better results. This is just a small example of the back-propagation, note that in bigger networks and with the Keras [Chollet et al., 2015] or the Tensorflow [Abadi et al., 2015], back-propagation works a bit differently. Interested reader is encourage to look at the Goodfellow et al. [2016].

2.4 Architecture

In our project, we focused on uniqe type of fully-convolutional neural network (FCN) – U-net Chen et al. [2018]. As shown in Figure 2.7, we can see that the network has U-shape. The first part of the network is down-sampling, and it is build up from blocks with two convolutions with filter size 3×3 and step 1. Afterwards, the 2×2 max-pooling is employed to down-sample the feature maps twice. Every convolution layer is followed by LeakyReLU activation function. In the bottom of the network, two convolution layers are used, and after them, up-sampling starts. Up-sampling layer consists of the block with single transpose convolution and two regular convolutions. After each of these operations, the activation function follows. The feature maps with same spacial size from down-sampling part are concatenated to the new feature maps from transpose convolution before regular convolution is done.

The input and output shapes are the same $256 \times 256 \times 1$. Although FCN has the advantage of being used on any input size during the training, we decided to use the same size during the training and evaluation. Default loss function is L_1 loss. For optimisation process we choose gradient descent algorithm based Equation 2.10 Adam optimiser [Kingma and Ba, 2014]:

$$m^t = b_1 m^{t-1} + (1 - b_1) g^t, \tag{2.17}$$

$$v^t = b_2 v^{t-1} + (1 - b_2)(g^t)^2, \tag{2.18}$$

$$g^t = \frac{\partial L}{\partial w_i}, \tag{2.19}$$

$$w^t = w^{t-1} - \frac{\alpha m^t}{\sqrt{v^t} + \epsilon}, \tag{2.20}$$

where m is a first moment vector, v is second moment vector, b_1 and b_2 are exponential decay rates for the moment estimates, g^t is the gradient of the lost function, α is the learning rate and ϵ is a small constant used to avoid zero division.

2.5 Training process

The length of training is counted in the epochs; one epoch refers to one cycle through all images from the data-set. In our project, the epoch has 150 iterations. During one iteration random part of each image is chosen and used as the input for the network. The loss function is calculated, and weights are updated after every iteration. The number of epochs depends on loss calculations; if loss is not changed for several epochs, and it is sufficiently

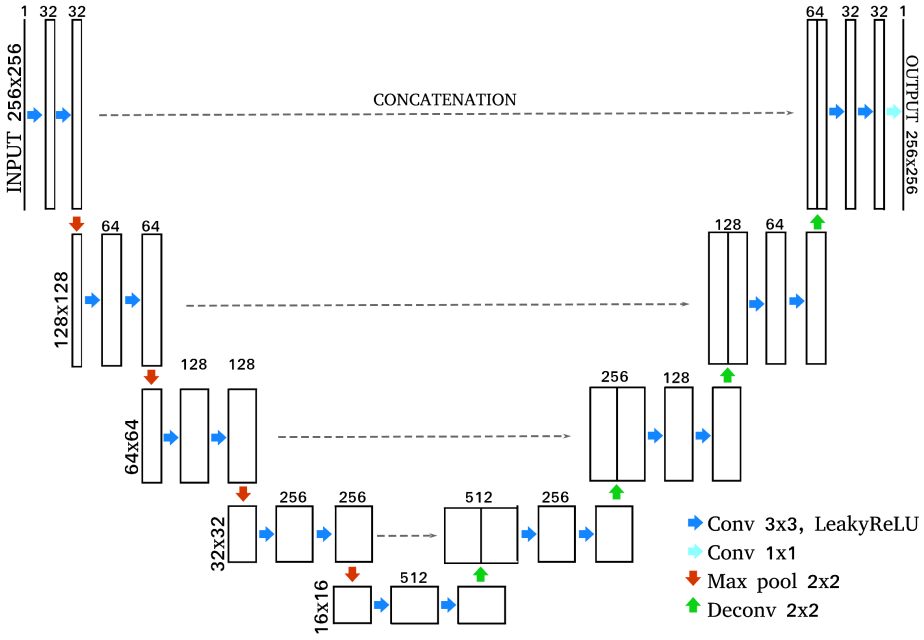

Figure 2.7: U-net architecture. Differently coloured arrows denote operations between layers. Numbers above represent numbers of feature maps in each layer. Numbers at the side of the images are width and high of the feature map. Dashes lines shows which layer from down-sample part are concatenated with layers in up-sample part. Notice that concatenates layers have the same width and high.

low, training can be stopped. The training of the neural network can be summarised as the algorithm:

1. Random weights initialisation.
2. Start of the epoch:
   (i) Take an image from the data-set,
   (ii) Pick a random position on the image, and crop 256×256 part of the image,
   (iii) Feed it into the network,
   (iv) Calculate the loss function,
   (v) Update weights,

(vi) Repeat (i)-(v) over whole data-set.

3. Repeat step 2 until loss converge.
4. Evaluate the network.
5. Win the Nobel prize for it (not you, me!)

2.6 Evaluation metrics

To evaluate the networks, we use several metrics. Standard metrics used to check the performance of the network employ for denoising problem are the Peak Signal-to-Noise Ratio (PSNR) and Structural SIMilarity (SSIM, Wang et al. [2004]). However, for our purpose, we have also to create metrics to examined the astrophysical properties of the images.

2.6.1 Common metrics

To compare the quality of noisy/denoised image to the original image PSNR is useful. In case of normalised images,which have the same min and max values, PSNR is always positive number. In our project, `FITS` files are used and in that case, min and max values are not the same for the image. That can result in negative PSNR. For identical images PSNR is indefinite as we can see from its definition:

$$\mathrm{MSE} = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j) - I'(i,j)]^2, \tag{2.21}$$

$$\mathrm{PSNR} = 10\log_{10}\left(\frac{\max(I)^2}{\mathrm{MSE}}\right), \tag{2.22}$$

where $m \times n$ is size of the images and $\max(I)$ is the maximum value of the original image. In many cases, also original image contains noise, and therefore denoised image can have higher quality and PSNR can be misleading.

SSIM is more sophisticated metrics and measure structure(s), contrast (c) and luminance (l) of the image:

$$l(x,y) = \frac{2\mu_x\mu_y + c_1}{\mu_x^2 + \mu_y^2 + c_1} \tag{2.23}$$

$$c(x,y) = \frac{2\sigma_{xy} + c_2}{\sigma_x^2 + \sigma_y^2 + c_2} \tag{2.24}$$

$$s(x,y) = \frac{\sigma_{xy} + c_3}{\sigma_x \sigma_y + c_3} \tag{2.25}$$

$$c_1 = (k_1 L)^2 c_1 = (k_2 L)^2 c_3 = \frac{c_2}{2} \tag{2.26}$$

where x and y denotes images (with same size), μ is average and σ is variance in respect to their index, L is the dynamic range of pixel values and k_1 and k_2 are small constant. Then SSIM is defined as:

$$\mathrm{SSIM}(x,y) = [l(x,y)^\alpha \cdot c(x,y)^\beta \cdot s(x,y)^\gamma] \tag{2.27}$$

where α, β, γ are weights and if we set them to one we get:

$$\mathrm{SSIM}(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}. \tag{2.28}$$

2.6.2 The source detection

To examined astrophysical properties of the images we decided to used the ratio of the signal (SNR or r), a number of detected stars and their flux. SNR is a crucial quantity for astronomical observation, it is the ratio between signal and noise. We calculate mean SNR for N_s stars by calculating their total flux F_i, flux error E_i and measuring background (B) of image:

$$r = \frac{1}{N_s} \sum_{i=1}^{N} \frac{F_i}{\sqrt{E_i^2 + B^2}}, \tag{2.29}$$

If we want to improve r we can make more images of the same part of the sky and stack them together. Because noise in the images is random, stacking will help to reduce it and increase the depth of the image. r_N of N stack images is higher than r_1 of a single image by a factor r_f:

$$r_f = \sqrt{N} = \frac{r_N}{r_1}. \tag{2.30}$$

In our work we are interested in: how many noisy images (N) we would need to stack together in order to obtain SNR of image generated (r_G) by the network:

$$r_f = \sqrt{N} = \frac{r_G}{r_N}. \tag{2.31}$$

Stars are also detected in real, noisy and generated images and then cross-match tables and divided detected stars into categories:

- *True positive* – stars detected on both, real image and compared image
- *True positive rate* – percentage of stars detected on both, real image and compared image

For True positive stars we compare flux from real F_R and noisy/generated image F_C and calculated δ_F:

$$\delta_F = \frac{F_R - F_C}{F_R}. \tag{2.32}$$

The relative δ_F is calculated for all stars and then mean value of the error is used to compare quality of the images. To detect stars Star Extractor or shortly SExtractor [Bertin and Arnouts, 1996] is used and to cross-match tables of detected stars `STILTS` [Taylor, 2006] is employ. Results are different, depending on parameters. Those are set to values `DETECT_MINAREA` = 10, `DETECT_THRESH` = 7, `BACK_SIZE` = 64, for rest of ones see link[1]. SNR is calculated from `FLUX_ISO` and corresponding error. To confirm the result for the best network, we also detected stars with the Munipack [Hroch, 2014].

2.6.3 The intensity distribution

The image intensity distribution is another aspect which is important to study. To test the distributions different tests are employ:

- Student's t-test
- Kolmogorov-Smirnov test
- Kullback–Leibler divergence

Student's t-test

The test measures whether the observed if difference in average values between the two groups is statistically significant. In our case, the two-sample t-test for independent samples is utilised. The null hypothesis of the test is that the means of two samples are equal in given dispersion σ_1, σ_2. The t-test starts by computing a test statistic on the two sets of observations. The t-statistic is defined as:

$$t = \frac{\bar{x_1} - \bar{x_2}}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}}. \tag{2.33}$$

[1] `https://github.com/Sponka/Astro-U-net/tree/master`

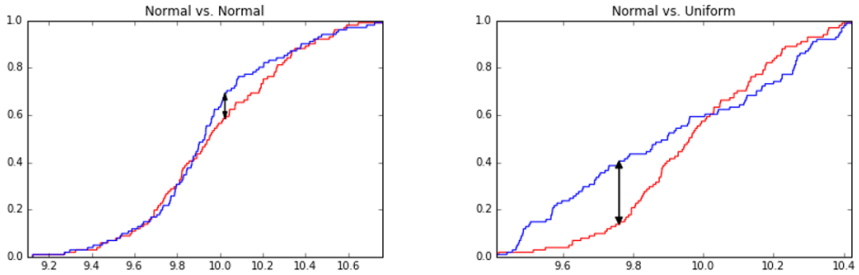

Figure 2.8: Left: CDF of two Normal distributions. Right: CDF of Normal and Uniform distribution. Image from Skiena [2017].

.

The $\bar{x_i}$, σ_i and n_i are the mean, standard deviation and size of sample i. The compared samples in our case have same size $n_1 = n_2 = n$ and we assume their standard deviations are also equal. For the t-statistic value there is the significance level, usually denoted as α. For a desired significance level and number of degrees of freedom (essentially the sample sizes), the table entry specifies the value v that the t-statistic t must exceed. If $t > v$, then the observation is significant to the α level. The p-value is probability that $|t|$ could be this large or larger just by chance, for distributions with equal means. Therefore, a small numerical value of the p-value ($0.001 or 0.01$) means that the observed difference is "very significant" (p-value from Press et al. [2007])

The Kolmogorov-Smirnov test

The Kolmogorov-Smirnov (KS) test compares the cumulative distribution functions (CDFs) of the two sample distributions and assesses how similar they are. The statistic is defined as:

$$D = \max_{-\infty \leq x \leq \infty} |F_1(x) - F_2(x)|, \tag{2.34}$$

where F_i is CDF of the first and second sample respectively. In Figure 2.8 is the statistic D shown as black arrow. The null hypothesis is rejected when two distributions differ at the significance level of α as:

$$D > c(\alpha)\sqrt{\frac{n_1 + n_2}{n_1 n_2}}, \tag{2.35}$$

for same sized samples:

$$D > c(\alpha)\sqrt{\frac{2}{n}}, \tag{2.36}$$

where the $c(\alpha)$ is defined as:

$$c(\alpha) = \sqrt{-\ln\left(\frac{\alpha}{2}\right) \cdot \frac{1}{2}}, \tag{2.37}$$

then we can write:

$$D > \sqrt{-\ln\left(\frac{\alpha}{2}\right) \cdot \frac{1}{2}} \cdot \sqrt{\frac{2}{n}}. \tag{2.38}$$

From Equation 2.38 it is apparent that when we want to confirm the null hypothesis (eg. $\alpha = 0.05$) for large sample (eg. $n = 1000$), the D statistic have to be low (~ 0.06).

The Kullback–Leibler divergence

The Kullback–Leibler (KL) divergence, or sometimes called relative entropy, is a measure how one distribution differ from the other, however, it cannot be used as distance between the samples. The KL divergence calculate how much information is lost when we approximate one distribution with another. When we want to find, which of our approximate distribution fit the target distribution better, the lower the KL divergence value, the better we have matched the true distribution with our approximation. It is defined as:

$$D_{KL}(p||q) = \sum_{i=1}^{N} p(x_i) \cdot \log \frac{p(x_i)}{q(x_i)}, \tag{2.39}$$

where p (original image) is the distribution which is supposed to be approximated by distribution q (generated/noisy image). It is important to noticed that KL divergence is not symmetrical, therefore:

$$D_{KL}(p||q) \neq D_{KL}(q||p). \tag{2.40}$$

Hubble space telescope data

Hubble Space Telescope is bringing us the light from distant worlds for over 30 years now. In 2009 Wide Field Camera 3 (WFC3) was installed to replace successful Wide Field Planetary Camera 2 (WFPC2) in the Hubble Space Telescope (HST). WFC3 has two channels – Ultraviolet-Visible channel (UVIS) and Infrared channel (IR). UVIS contains together 62 filters in wide-, medium- and narrow-band. Range goes from near-UV (200 nm) to the near-IR (1700 nm). WFC3 provide high-resolution, high-sensitive, wide-field data with a broad wavelength range.

It is not possible to observe IR and UVIS simultaneously during the observation. The optical and mechanical layout of the instrument is shown in the schematic diagram in Figure 3.1.

In our work, we focus on the WFC3 instrument UVIS. Instrument field of view is 162×162" from $200 - 1000nm$ with a plate scale of 0.040"/pixel. The channel use mosaic of two 4096×2051 pixels back-illuminated CCDs with size approximately 6×6 cm. There is small gab between CCDs ~ 31 pixels which is ~ 1.4". The exposure time limitation is $0.5seconds$. The depth of the CCD full well is ~ 70000 electrons in every pixel.

Filters of the UVIS instrument have been chosen to cover a wide variety of scientific interest as photometry of stellar sources, imagining of nebulae or colour selection of distant galaxies, et cetera... Also, some of the filters match the most commonly used filters from WFPC2 and the Sloan Digital Sky Survey (SDSS). In our work, we pick two wide-band filters (from 12 of them), a wide V-band filter (F606W) and one of the Johnson-Cousins BVI set (F555W), where both of them match two of WFPC2 filters. The effective wavelength of F606W and F555W filters are 530.8 and 588.7 nm and width of filters are 156.2 and 218.2 nm respectively. We have chosen these filters because their width is the biggest one from all wide-band filters. The amount of images captured in filters F606W and F555W is high, and they have significant overlap, as shown in Figure 3.2. Measurements of properties of filters are done in a laboratory at the temperature of 20°C. However, in space, they are operating at 0°C so that this difference may cause a wavelength shift.

As predicted in models of the UVIS detector, UVIS can exhibit different types of ghost (artefacts) due to reflection. Our data also contains some ghosts, and it is not in the goal of our work to remove them. For more information see Dressel and Gennaro [2018].

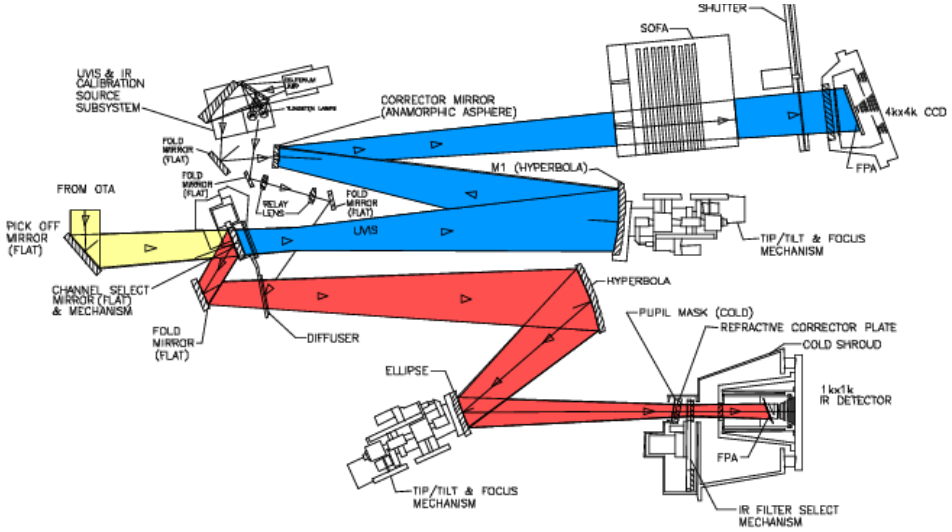

Figure 3.1: Image shows a schematic diagram of the instrument's optical and mechanical layout. From the HST Optical Telescope Assembly (OTA) light is going to WFC3 pick-off mirror (POM) where is reflect to the instrument. With a channel-select mechanism (CSM) it can be decide if light eill go into UVIS channel or IR channel.

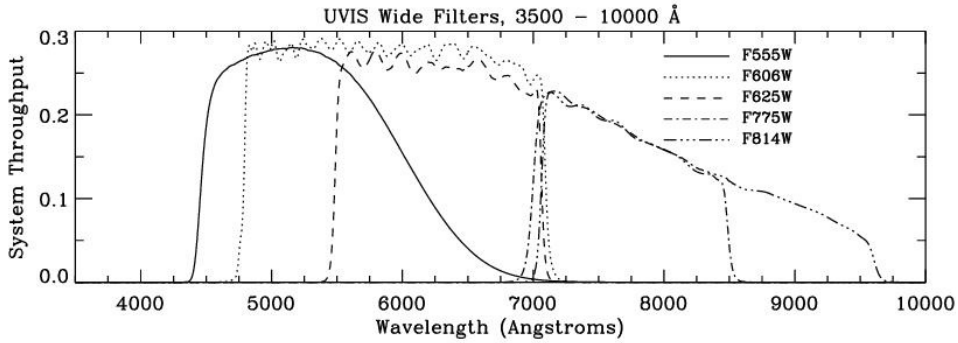

Figure 3.2: Throughput of the WFC3 UVIS wide-band filters, our target are optical filters are F555W (full line) and F606W (dotted line).

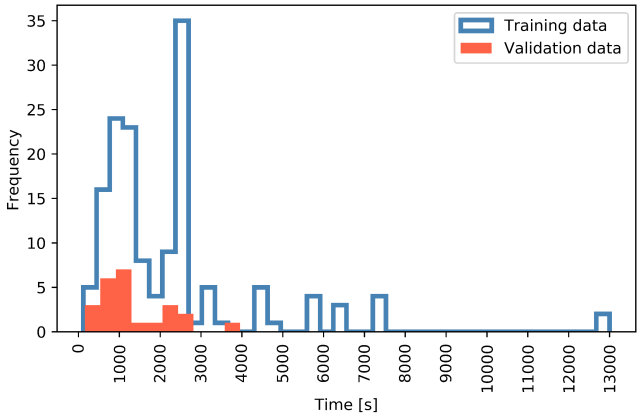

Figure 3.3: Exposure time of ground truth images in the training and the validation data-set.

3.1 Training data

For training of ASTRO U-net, we went by hand trough the WFC3 F555W and F606W data and picked 156 of them – 150 for training the network and 6 to evaluate it. The data contains various astronomical objects like nebulae, galaxies, stars, clusters, etc. Images have different exposure time, as shown in Figure 3.3. These *real data* are used as *ground truth* for the network, images which are compared with the output of the network. The network input are *simulated data* – real data with additional noise (Subsubsection 3.1.1). To avoid loss of the information, we used FITS files for the training, and all images are in *electrons/second*. To visualise images in this work, we use ZScaleinterval and Linear stretch.

3.1.1 Simulated data

Astronomical images are degraded by different types of noise as instrumental noise (dark noise, read-out-noise) or photon noise. The training data consist of two types of images:

- Input data – short exposure images with added noise
- Ground truth – long exposure images which are used for comparison of the network output images to calculate the loss function

Photons emitted from any incoherent source have an inherent statistical variation and are distributed according to a Poisson distribution. The noise

associated with the number of photons and randomness of its arrival to the detector is *photon noise* (N_{photon}) – it is directly dependent on the number of photons recorded by the measuring device. Dark noise (N_{dark}) is caused by thermally excited electrons of the CCD, and it is strongly correlated with temperature and independent from light photons captured by the detector. Because of radiation damage from the Earth radiation belts, the dark current (DC) is slowly increasing every year by $\sim 0.5\text{e}^-/\text{hr/pix/year}$. In 2016 value of the dark current of UVIS detector was $7\text{e}^-/\text{hr/pixel}$. During the read-out, the information from CCD chip captured electrons are converted into a voltage by amplifiers (UVIS has four amplifiers). This electronic device brings read-out-noise (RON) into images, and its value is ∼three electrons. Both dark and read-out noise has Poisson distribution $P(\mu)$.

A common long exposure time image is sum I_{long} of detected photons (S) and noise (N, in electrons):

$$I_{\text{long}} = S + N. \tag{3.1}$$

The noise consist of many different types of noise, and in our simulation we will work with three types of noises mentioned above:

$$N_{\text{photon}} = P(\sqrt{S}), \tag{3.2}$$

$$N_{\text{dark}} = P(\sqrt{I_{\text{DC}} \cdot t}), \tag{3.3}$$

$$N_{\text{ron}} = N\left(0, \sigma_{ron}\right), \tag{3.4}$$

where I_{DC} is a dark current and t is the exposure time of the image. The total noise is:

$$N_{\text{total}} = N_{\text{photon}} + N_{\text{dark}} + N_{\text{ron}} = I_{\text{short}} + P(\sqrt{I_{\text{short}}}) + P(\sqrt{I_{\text{DC}} \cdot t}) + N\left(0, \sigma_{ron}\right). \tag{3.5}$$

To create short exposure time image I_{short} we divide long exposure one by exposure time ratio (r), which is ratio between exposure time of I_{long} and I_{short}:

$$I_{\text{short}} = I_{\text{long}}/r. \tag{3.6}$$

In order to obtain the input images we add total noise into short exposure image:

$$I_{\text{input}} = I_{\text{short}} + N_{\text{photon}} + N_{\text{dark}} + N_{\text{ron}}. \tag{3.7}$$

To create input data, also called noisy images or images with artificial noise the Equation 3.1 – Equation 3.7 are use to built up a code. The part of the python code, which is used to produce simulated data noisy images is:

```
"""exp_time - exposure time of the original image
   ron - read out noise - from the Hubble handbook
   dc  - dark current - from the Hubble handbook"""
ratio  = 2    # exposure time ratio
width, height = original_image.shape[0:2]
short = original_image/ratio
# Dark noise
DN = np.random.poisson(np.sqrt(dc*exp_time/ratio), (width, height))
# Read-out noise
RON = np.random.poisson(ron, (width, height))
# Photon noise
SN = np.random.poisson(np.sqrt(np.abs(short))
noisy_image = (short + SN + RON + DN)
```

The code to download the data-set and produce simulated data is available in the Appendix. We are fully aware of the differences between simulated data and real data. We chose this option because, with the Hubble archive, it is not straightforward to fulfil our requirements for training data. To demonstrate the difference between real data and simulated, we picked two images of Seyfert 2 Galaxy ESO 55-2 with 20 seconds exposure time and combined them to obtain one longer exposure time image. As we can see in the Figure 3.4, the pixel distribution of the shorter image is wider than the distribution of longer exposure time image. Distribution of simulated images with a ratio more that one is also more extensive.

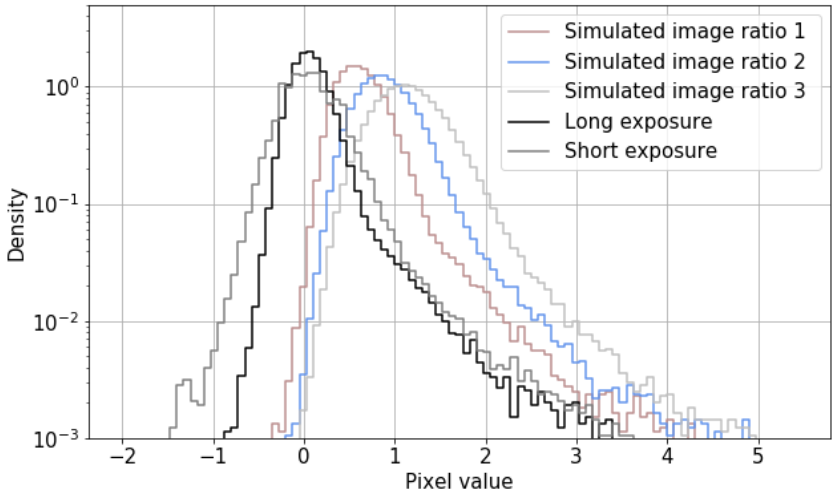

Figure 3.4: Pixel distribution of part of the image which contains the object

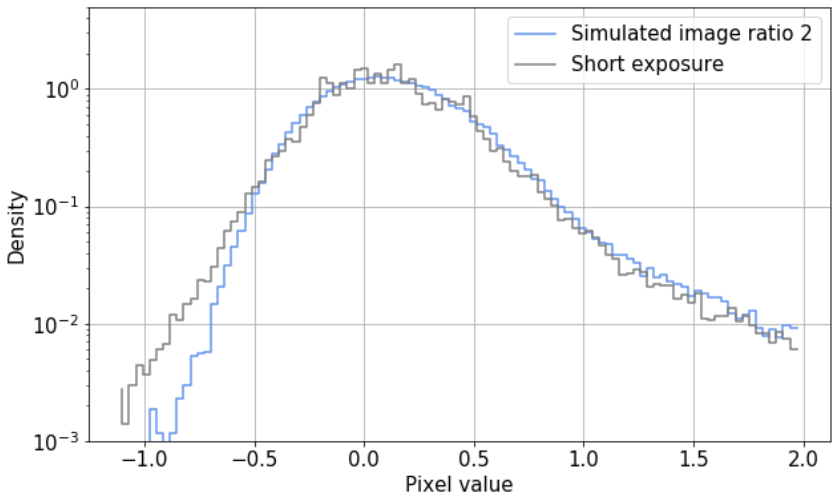

Figure 3.5: Pixel distribution from same region as at Figure 3.4. The distribution of simulated image is shifted by the difference of mean of both distribution.

3.1.2 Output data

The network input has size 256×256, but all of our images exceed this input size. To create the output image, we employ the mosaic approach. First, the zero-padding is added to the image – to ensure, that every part of the image is generated same times. Then, image is cut to the pieces as – at the position i, j the cut with kernel the size 256×256 and then the kernel moves to the position $i + 32, j + 32$ and the same sized cut is done, et cetera. (Figure 3.7). After that, all cuts serves as the input for the network and then return into the original position in a new array, sum up and average. This method does not change the resulting image significantly, although it helps to smooth check-board artefact created by the transpose convolutional layer [Odena et al., 2016].

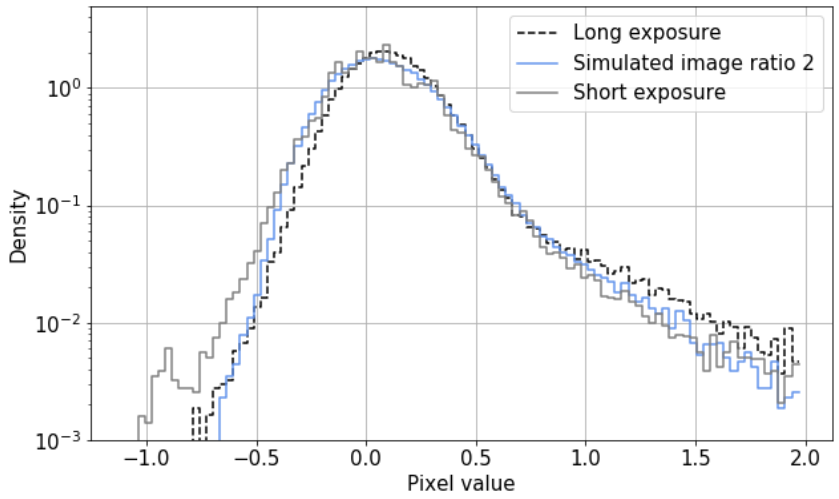

Figure 3.6: Pixel distribution from same region as at Figure 3.4. The distribution of simulated image is shifted by the difference of mean and both distribution of simulated and short exposure image are scaled by square root of ratio two. As expected distribution almost completely overlap.

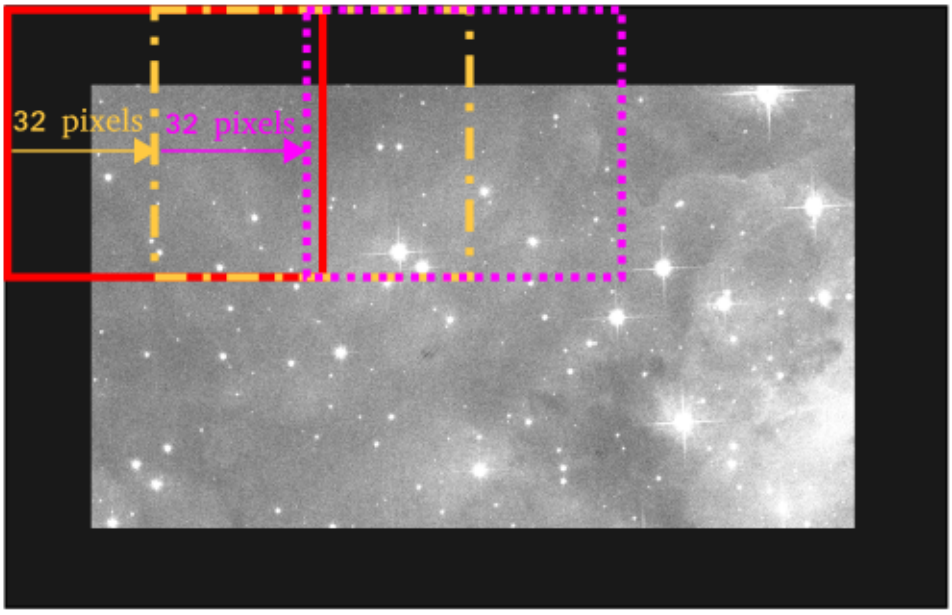

Figure 3.7: The mosaic method to create new image. After every cut, the kernel is moved 32 pixels as shown on the image.

Results

4.1 Experiments

We trained over 30 networks with different parameters to find the best ones during our study. In the following section, different experiments are described, and for the best network, deeper analysis is made. In every experiment, we used architecture from Subsection 2.4, input size of the images were 256×256 because of computational cost. The results of the experiments can be seen in Table 4.1. The architecture is inspired by [Chen et al., 2018] and written in Tensorflow [Abadi et al., 2015]. The networks are trained on the `NVIDIA GTX1080 Ti` of project Sandy in ESTEC.

4.1.1 Loss function

Subsubsection 2.3.3 describes different types of loss functions – L_1, L_2 and perceptual loss. Results of experiments showed that L_2 and perceptual loss is not suitable for our task, they do not improve the flux error of the network. Moreover, L_2 loss can easily get stuck in a local minima, as shown in Zhao et al. [2017]. They also showed that L_1 loss reaches a local minima more efficiently. Therefore L_1 is selected as the default loss function for every other experiment. This experiment (Experiment 1) achieved 84% of True positive rate, flux error 16%, PSNR 18 dB and SNR is 1.11.

We decided, after many different experiments, to make one more loss function experiment (Experiment 6) – we combine L_1 loss with Kullback-Leibler divergence. Results showed that KL divergence does not improve the performance of the network.

4.1.2 Number of input channels

As is mentioned in the introduction of this chapter, the size of the input images is 256×256. However, we want to experiment with the third dimension – the number of the channels. From all filters of the Hubble Space Telescope two are chosen – F555W and F606, and we want to see if the results will change if we put these filters in different channels. For F555W the first channel is used and for F606W the second one. We find that there is no difference in this approach and therefore, we do not consider separation of filters beneficial for our project.

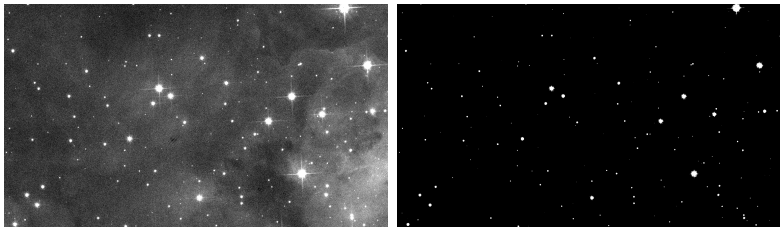

Figure 4.1: Example of segmentation map. The image on the right site is the segmentation map of the image on the left side. The segmentation map was created by star extractor.

4.1.3 Segmentation map

In Experiment 2, we have added the segmentation map of the input/output image into the second channel of the input/output. The segmentation map comprises from sources separated from sky background. We obtain these maps from Star Extractor. The main idea is to assign more importance to the object in the segmentation map because we want to recover the detected object more accurately. The segmentation maps helped to improve result of True positives – 98.9%, flux error – 3.9, PSNR – 20.3 dB and SNR – 1.58.

4.1.4 Exposure time ratio

In these sets of experiments, we are interested in how adding exposure time ratio into network change the result. Hence we try different options: adding ratio into the bottom of the network or multiplying images by it.

In Experiment 3, the network has input with image and segmentation maps in separated channels and one channel in the output. The exposure time ratio is added into the bottom of the U-net, and it is trained for 4000 epochs. We train the same network without segmentation map – Experiment 4. The flux error of Experiment 4 is slightly higher, and we have less true positives; PSNR was the same although the training time is shorter – 3000 epochs.

In Experiment 5, we multiplied the input image and the segmentation map by exposure time ratio. The network is trained for 5000 epochs, and in comparison with previous experiments (3,4), the network is trained for more epochs, and it has no significant impact on the results. After this, we conclude that in the next experiments, the exposure time ratio will be added at the bottom of the network.

Next to investigate are multiple exposure time ratios – we trained the

networks with higher exposure time ratios. We used set up from Experiment 4, and during the training of Experiment 7, the ratio is randomly chosen from 2 to 5, this means, that during the random training crop with random exposure time is feed into the network. In contrast with Experiment 8, the ratio was not selected randomly. During one iteration, the network sees the same crop from the image with all ratios from 2 to 5. After every ratio, the weights of the network are updated. This approach gives us better results than one with randomly selected ratios.

All the following experiments are done with ratio two added into the bottom of the network.

4.1.5 Activation function

As a baseline for our project Leaky Rectified Linear Unit (LeakyReLU) activation function is chosen; however, we want to try what market with activation functions offer. All previous experiments are done with LeakyReLU one. In the following experiments, settings from Experiment 4 are employ. Experiment 9 – ReLU activation function results in higher flux error then with LeakyReLU. Conversely, ParametricReLU (PReLU) in Experiment 10 improved flux error and SNR, though the network is trained longer, for 4000 epochs. Also, the PReLU activation function is adding almost 3000 trainable parameters. The parameter a is high in the first layers and some of the last layers and in every other, it is near zero (Figure 4.2, Table 4.2). In the last, 11th experiment, the self-gated activation function (SWISH) is used and it helps to improve the SNR and the flux error. All activation functions are describe in Subsubsection 2.3.2.

Experiment	True positive rate [%]	Flux error [%]	PSNR [dB]	SNR_f
1	84.0	16.9	18.0	1.11
2	98.9	3.9	20.3	1.58
3	99.0	3.5	20.4	1.60
4	98.9	3.8	20.3	1.60
5	98.8	3.3	20.4	1.59
6	97.9	5.0	19.1	1.51
7	98.7	6.9	18.8	1.57
8	98.4	4.9	19.4	1.60
9	98.9	4.0	20.4	1.56
10	99.0	3.2	20.3	1.60
11	99.0	2.8	20.1	1.65

Table 4.1: Comparison of the networks experiments. The flux is measured by the Star Extractor.

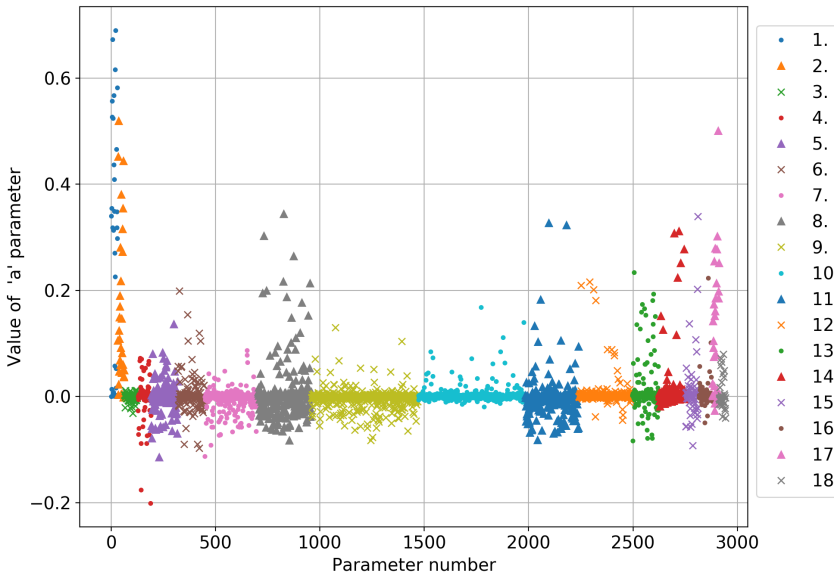

Figure 4.2: Trainable a parameter from ParametricReLU activation function

Layer	Number of feature maps	Median $[\cdot 10^{-3}]$	Mean $[\cdot 10^{-3}]$
1	32	318.18	292.03
2	32	98.91	151.51
3	64	−0.24	−1.23
4	64	3.82	−4.39
5	128	0.16	−0.52
6	128	0.14	4.5
7	256	−1.18	−6.5
8	256	0.12	5.29
9	512	−0.3	-3.51
10	512	0.26	3.09
11	256	−1.15	−1.54
12	256	0.84	5.89
13	128	1.55	16.36
14	128	4.62	17.93
15	64	2.5	10.8
16	64	1.21	7.5
17	32	85.54	117.55
18	32	−2.79	6.52

Table 4.2: In experiment 10, the parametric ReLU activation function is employ. The table shows information about the trainable parameter a in different layers.

4.2 Network Evaluation

As the baseline network, we choose one with the best results presented as Experiment 4. The loss function of the network is $L1$ loss, it has LeakyReLU activation, and the exposure time ratio is added into the bottom of the network. The network is trained for 37 hours or 3000 epochs on `NVIDIA GTX1080 Ti`. It takes 30 seconds to generate an image with size 1200×1400 on the GPU. Before we start the evaluation of the network, we would like to remind our readers about the meaning of used terms:

- Ground truth, original image, comparison image – refers to an image which is downloaded from HST archive. This type image also serves as a comparison for our test – the goal of the network is to generate a similar image.
- Input image, noisy image, the image with noise, simulated image – this

image is the input for the network. It is created from the original image by adding artificial noise (Subsubsection 3.1.1)– it is supposed to simulate image with short exposure time

- Generated image, output image – image generated (created) by the network

4.2.1 Source detection

One of the most crucial evaluation techniques used is a comparison of star flux. The original images are compared with both the noisy and generated ones. For the evaluation, the Star extractor [Bertin and Arnouts, 1996] and Munipack [Hroch, 2014] are employed.

The Table 4.4 and 4.5 show tables containing information about measured flux from 22 tested generated and noisy images respectively, where stars are detected by the Munipack. The Figure 4.4 and Figure 4.8 show the flux error versus the star flux from image generated by the network. Stars are detected

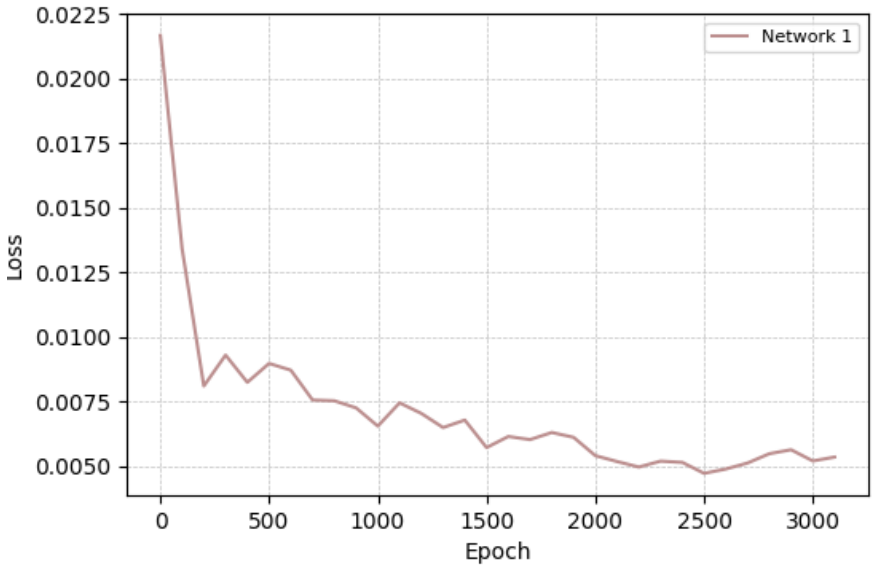

Figure 4.3: Network 1 training loss

Method	Image type	Aperture size [pixel]	Mean error [%]	Median error [%]	True positive rate [%]
Munipack	G	1	3.6	1.7	92.1
Munipack	N	1	3.8	3.0	64.9
Star Extractor	G	1	6.5	4.7	95.4
Star Extractor	N	1	6.0	3.9	57.8
Star Extractor	G	10	3.5	2.0	95.4
Star Extractor	N	10	4.8	4.1	57.8

Table 4.3: The network experiments results. Image type can be G for the generated image or N for the noisy image. The mean and median error (star flux error) are mean of all tested images.

at the images shown in the Figure 4.6 and Figure 4.10 respectively.

The Table 4.6 and 4.7 show 9 tested generated and noisy images respectively by Star Extractor. Tables contains flux error for apertures with size 1 and 10 pixels, where first error correspond to size 1 and the second error to size of aperture 10. The Figure 4.7 and Figure 4.7 shows the detected stars with the star flux error denoted by the coloured circle.

The Table 4.4 and Table 4.6 show the result of measured flux is generated by Munipack and Star Extractor. The mean results for both methods are in Table 4.3. We find that error for different sized apertures varies; the median error is 1.7% and 4.7% for Munipack and Star Extractor respectively for aperture with size 1 pixel and 2.0% for Star Extractor with aperture size 10 pixels. The mean (median) error can be higher for the generated images than for the noisy images in Table 4.5, because the faint stars have the highest error as seen in Figure 4.5 and 4.9 are not detected. For both methods, it is obvious that there is a lack of detection in noisy images for lower fluxes. The true positive rate is higher for generated images by both star detection methods.

This test shows that the network is able to recover stars with low star flux error compared to the original images. Moreover, the number of detected stars is higher by $\sim 30\%$ than on the noisy images. Note, that to detect stars in all type of the images same sky threshold is used. This suggests that the image noise is lower in the generated images. If we decrease or increase the threshold, the results will vary.

Image	Mean error [%]	Median error [%]	True positive	True positive rate [%]
1	1.4	0.6	265	97.4
2	2.1	1.1	252	96.2
3	6.1	2.9	171	91.0
4	8.9	6.3	158	70.5
5	9.3	4.2	224	68.3
6	5.8	2.0	618	97.6
7	4.7	1.7	728	96.8
8	4.6	1.5	2201	97.6
9	3.9	1.5	2606	94.8
10	4.5	2.9	15835	98.8
11	1.0	0.4	62	95.4
12	4.9	1.5	7146	97.8
13	2.3	1.1	17234	98.9
14	2.1	0.6	156	85.2
15	0.6	0.3	6733	97.5
16	2.7	1.1	615	91.4
17	1.2	0.6	44	91.7
18	7.9	2.7	688	82.1
19	2.1	1.3	2119	96.2
20	1.5	0.8	2366	96.9
21	1.0	0.5	4892	99.1
22	1.1	0.7	59	92.2
Mean	3.6	1.7	2960	92.1

Table 4.4: The results of 22 generated images from the validation dataset. Stars are detected, and flux is measured by Munipack. For cross-match of tables, the Topcat is used, and the maximum positional difference is 1 pixel. The mean/median error refers to star flux error.

Image	Mean error [%]	Median error [%]	True positive	True positive rate [%]
1	3.3	2.5	238	87.5
2	3.2	2.3	202	77.1
3	4.7	3.6	125	66.5
4	4.2	3.6	38	17.0
5	5.0	3.9	100	30.5
6	5.8	4.4	463	73.1
7	5.5	4.3	547	72.7
8	4.7	3.7	1643	72.9
9	4.7	3.6	1975	71.9
10	5.8	4.6	10981	68.5
11	1.6	1.4	62	95.4
12	5.0	3.9	5790	79.2
13	4.7	3.6	15554	89.2
14	2.9	1.9	133	72.7
15	1.5	1.4	5103	73.9
16	2.9	1.8	434	64.5
17	1.4	1.4	29	60.4
18	4.8	3.3	373	44.5
19	3.9	3.0	858	38.9
20	3.2	2.6	1060	43.4
21	2.4	2.2	3109	63.0
22	2.5	2.1	41	64.1
Mean	3.8	3.0	2220	64.9

Table 4.5: The results for 22 noisy images. Stars are detected, and flux is measured by Munipack. For cross-match of tables, the Topcat is used, and the maximum positional difference is 1 pixel. In comparison with Table 4.4 we can see, that true positive rate is always lower, that signify Munipack detect fewer stars on the noisy images. The mean/median error refers to star flux error.

	Aperture 1 pixel		Aperture 10 pixels			
Image	Mean error [%]	Median error [%]	Mean error [%]	Median error [%]	True positive	True positive rate [%]
1	4.9	3.6	2.1	0.9	195	95.1
2	4.2	3.2	1.5	0.7	161	99.4
3	7.0	5.2	4.5	1.7	78	94.0
4	10.1	8.6	8.6	6.9	76	78.4
5	7.6	5.0	5.9	2.8	78	95.1
6	5.4	4.2	2.4	1.3	379	99.5
7	5.3	4.0	2.2	1.2	464	99.4
8	5.2	4.2	2.2	1.2	1370	99.0
9	5.1	3.9	2.3	1.1	1532	98.3
Mean	6.1	4.7	3.5	2.0	481	95.4

Table 4.6: The flux error of generated images detected by Star Extractor. The images correspond to the first nine from the Table 4.4. The flux is measured with aperture size 1 and 10, which corresponds to the first and the second mean and median errors, respectively. The size of the aperture does not influence the number of detection. The mean/median error refers to star flux error.

	Aperture 1 pixel		Aperture 10 pixels			
Image	Mean error [%]	Median error [%]	Mean error [%]	Median error [%]	True positive	True positive rate [%]
1	5.0	3.8	3.0	2.5	143	69.8
2	4.4	3.1	3.4	3.0	123	75.9
3	6.4	3.6	4.5	3.8	42	50.6
4	8.8	4.9	6.8	5.1	20	20.6
5	7.0	4.6	4.2	3.7	38	46.3
6	5.2	3.9	5.4	4.9	230	60.4
7	5.7	3.8	5.8	5.3	302	64.7
8	5.7	3.8	5.3	4.4	895	64.7
9	5.7	3.6	4.9	4.3	1049	67.3
Mean	6.0	3.9	4.8	4.1	316	57.8

Table 4.7: The flux error of noisy images detected by Star Extractor. The images correspond to the first nine from the Table 4.5. The flux is measured with aperture size 1 and 10, which corresponds to the first and the second mean and median errors, respectively. The size of the aperture does not influence the number of detection. The mean/median error refers to star flux error.

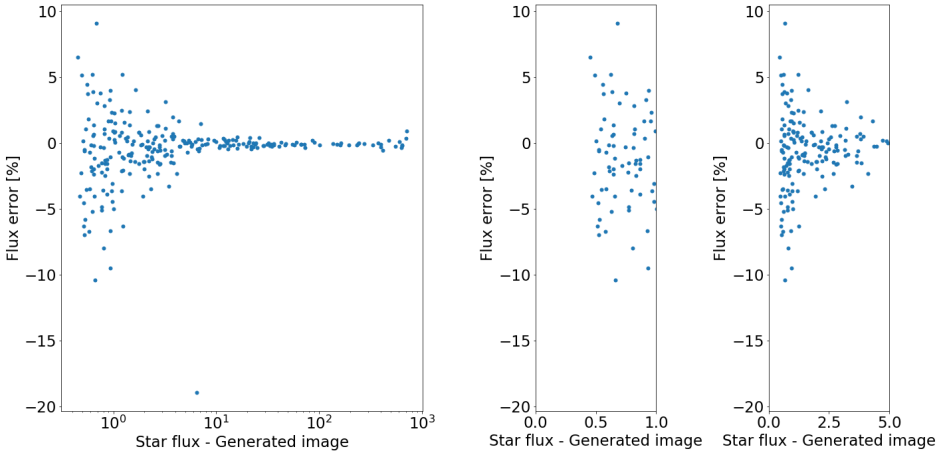

Figure 4.4: Flux error versus star flux for the second generated image in the Table 4.4; the image is generated by the network. Stars are detected, and flux is estimated by Munipack. The stars detection are from the generated image shown in the Figure 4.6.

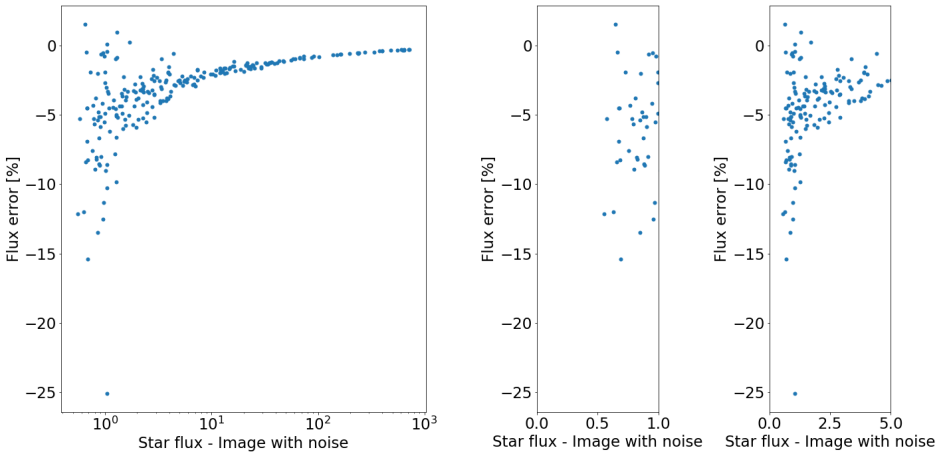

Figure 4.5: Flux error versus star flux for the second image with artificial noise from Table 4.5. Stars are detected, and flux is estimated by Munipack. In comparison with the same image generated by the neural network (Figure 4.4), it is apparent that with the network, we have more defections in the lower star flux region.

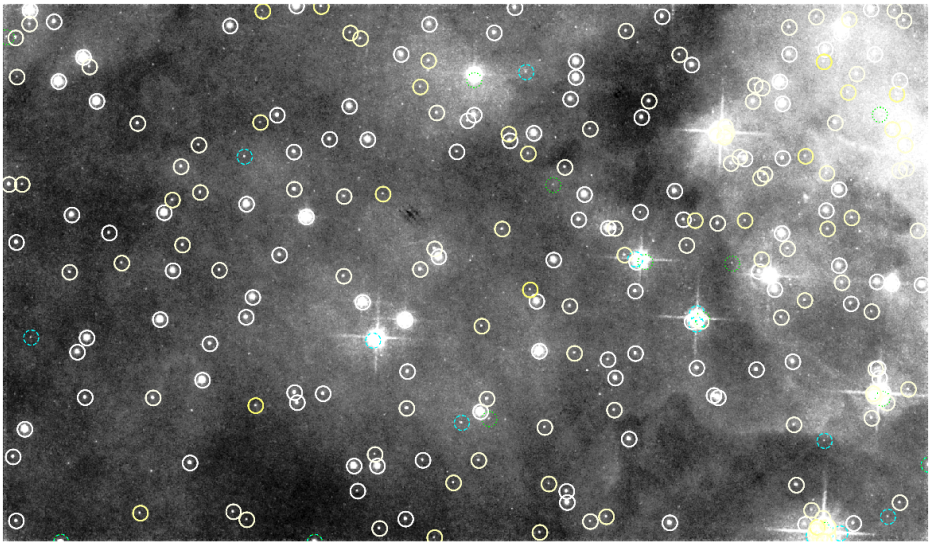

Figure 4.6: Stars detection on the second image from Table 4.4 for Munipack detection. The mean error is 1.4%, the median error is 0.6%, and the true positive rate is 97.4%. The green dotted circles are stars detected on the original image but missed on the generated; the blue dashes circles are stars detected just on the generated image. The full circles represent true positive stars, and the colour of the circle indicates flux error. In comparison with the same image with detection done by Star Extractor Figure 4.7, it is visible, that Munipack is able to detect (and cross-match) more sources.

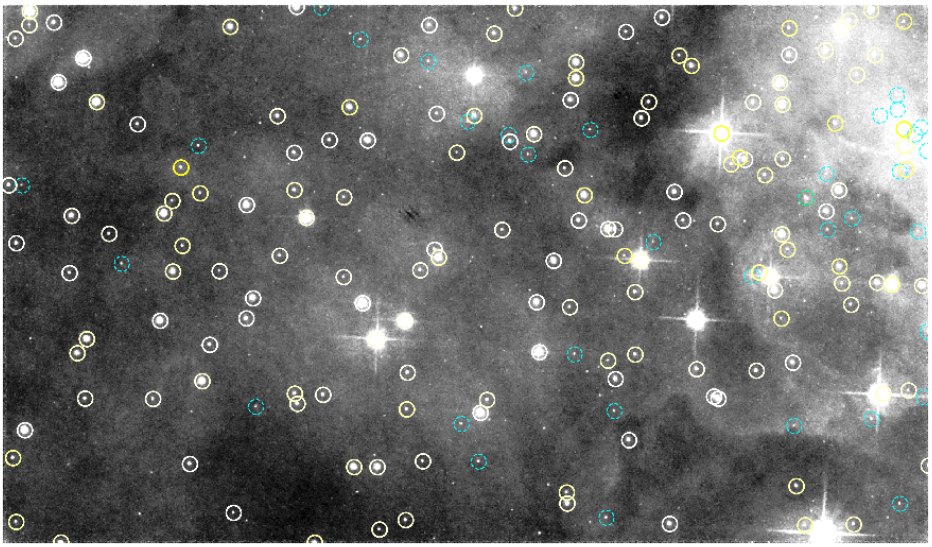

Figure 4.7: Stars detection on the second image from Table 4.6 for Star Extractor detection. The results for aperture 1 are the mean error 4.2%, the median error 3.2% and for the aperture 10 the mean error 1.5%, the median error 0.7% and the true positive rate for both apertures is 99.4%. The green dotted circles are stars detected on the original image but missed on the generated; the blue dashes circles are stars detected just on the generated image. The full circles represent true positive stars, and the colour of the circle indicates flux error. Many sources are not detected on the original image; on the other hand, they were detected on the generated image. If we compare those sources with sources detected at Figure 4.6, we observe that Munipack does not miss most of them.

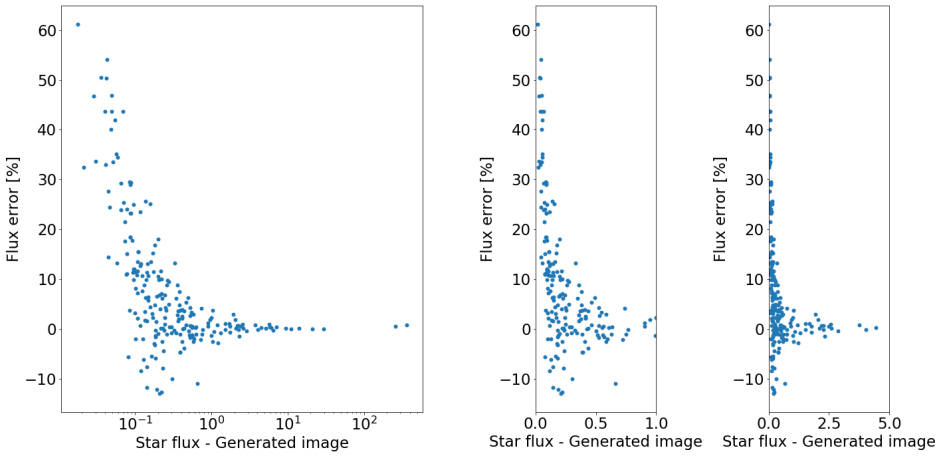

Figure 4.8: Flux error versus star flux for 5. generated image from Table 4.4. Stars are detected, and flux is estimated by Munipack. The generated image is shown in the Figure 4.6. The stars detection are from the generated image shown in the Figure 4.10

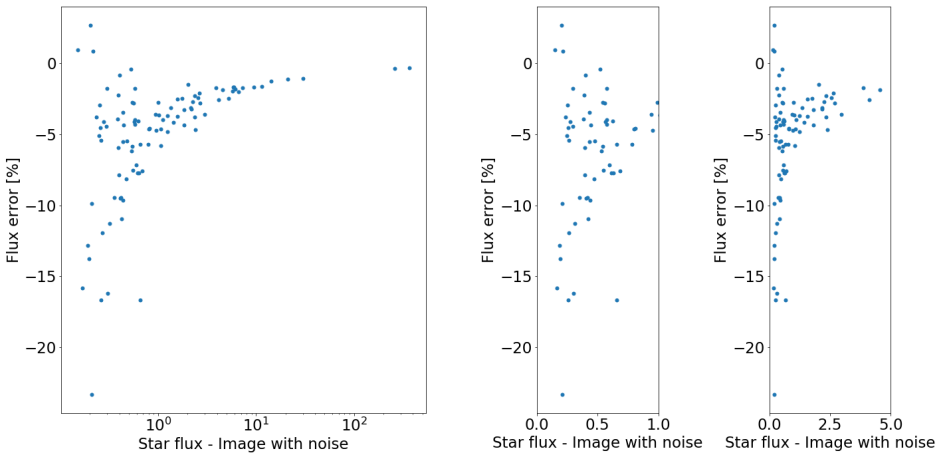

Figure 4.9: Flux error vs star flux for 5. noisy image from Table 4.5. In comparison with Figure 4.8 it is visible that there are much fewer detection of stars with flux smaller than 0.25. Stars are detected and flux is estimated by Munipack.

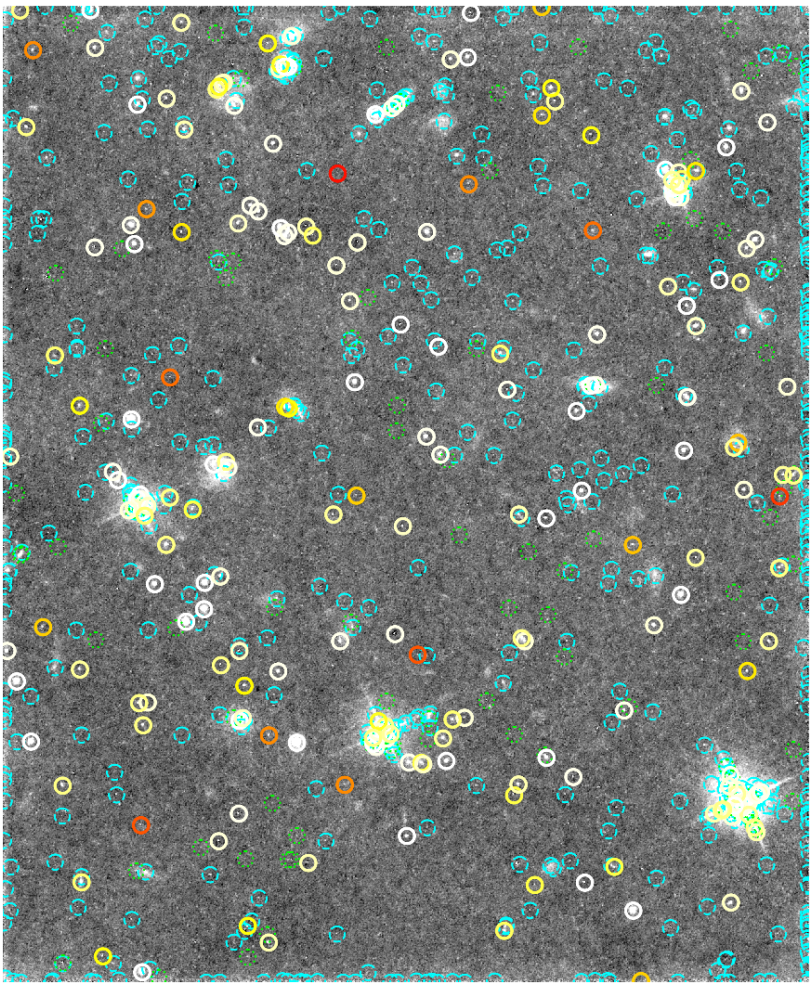

Figure 4.10: Stars detection on the 5. image from Table 4.4 for Munipack detection. The mean error is 9.3%, the median error is 4.2%, and the true positive rate is 68.3%. The green dotted circles are stars detected on the original image but missed on the generated; the blue dashes circles are stars detected just on the generated image. The full circles represent true positive stars, and the colour of the circle indicates flux error. It is apparent that Munipack detected a lot of false sources/cosmic rays on the original image (green circles). At the generated image there are a lot of false detection on the edge of the image (blue circles).

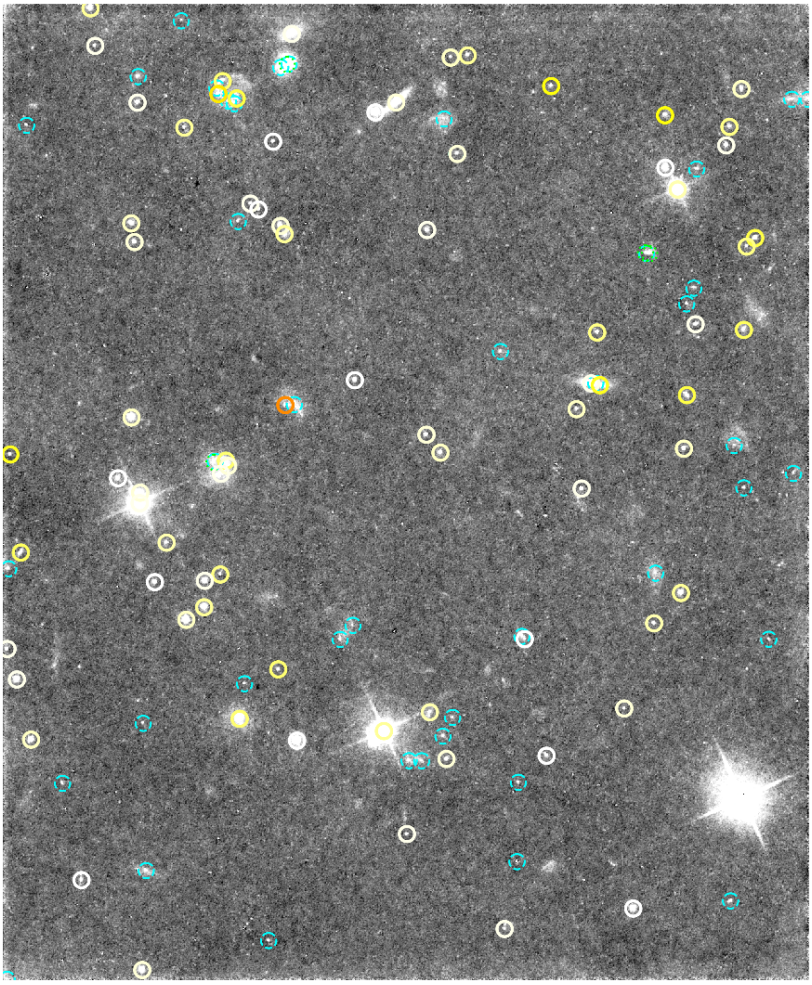

Figure 4.11: Stars detection on the 5. image from Table 4.6 for Star Extractor detection. The results for aperture 1 are the mean error 7.6%, the median error 5.0% and for the aperture 10 the mean error 5.9%, the median error 2.8% and the true positive rate is 95.1%. The green dotted circles are stars detected on the original image but missed on the generated; the blue dashes circles are stars detected just on the generated image. The full circles denote true positive stars, and the colour of the circle indicates flux error.

4.2.2 Randomness in the flux recovery

There is an important question: *Is the flux of the star recovered randomly?* Unfortunately, the answer to this question is not *42*. Luckily experiment to answer the question is simpler than the Deep Thought experiment. The experiment is performed on two images, and it goes as:

1. Pick small sub-window of image
2. Add simulated noise
3. Feed the image with noise to the network
4. Use Star Extractor to estimate the star flux
5. Repeat steps 2-4 ten times
6. Compare fluxes of the same stars in different generated images

The image is shown in Figure 4.12 was feed into the network ten times, every time with different random simulated noise. In Figure 4.13 we can see error of 20 stars on ten images. Star error was calculated by Equation 2.32. The stars with lower fluxes have more variable error than stars with high fluxes; a mean star flux error is $\sim 3.3\%$. The Figure 4.15 and Figure 4.16 shows spread of flux from generated image around original flux value. The grey dashed line shows the error estimated by Star Extractor. All generated stars are within the error bars.

The second studied image in Figure 4.17 shows the same trend as the first one. Low star fluxes are more variable than the stars with higher flux, although they do not exceed the error assign by Star Extractor. The mean star flux error is $\sim 2.1\%$. From Figure 4.19 and Figure 4.20 one can observe that higher fluxes are less variable, the spread around the original flux is smaller.

This experiment shows that there is some variability within flux recovery; nevertheless, the variability is still in boundaries of the estimates star flux error from the original image.

Near edges stars can have worse results. This can be caused by two factors: the training data contained near-edge artefacts, which the network could learn or the Star Extractor has a problem during the detection.

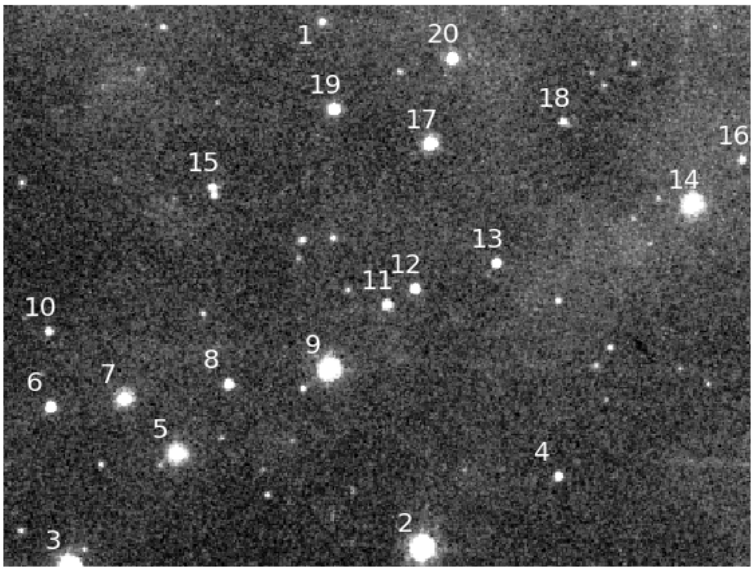

Figure 4.12: Sub-window image contains twenty stars. The star number is associated with star fluxes in Figure 4.13

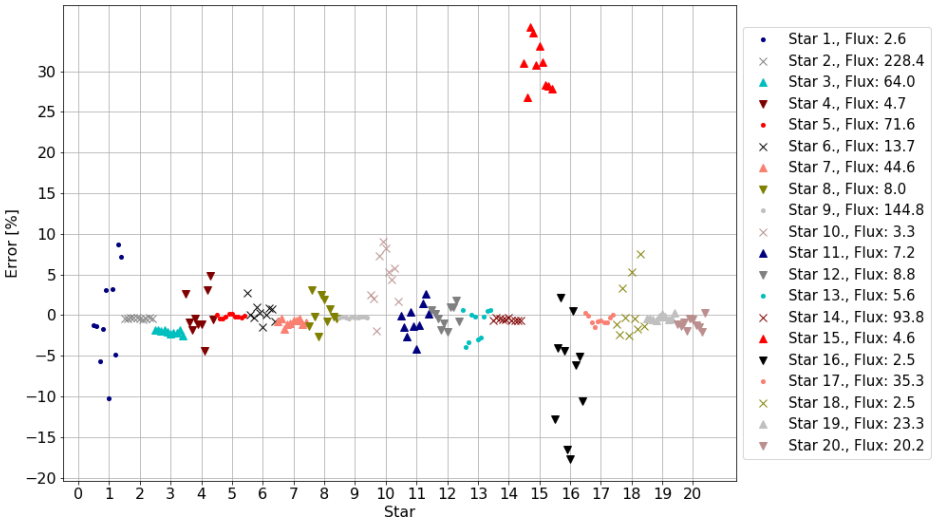

Figure 4.13: The star flux error of twenty stars from ten times simulated image Figure 4.12. The star error was calculated according Equation 2.32. Same stars have similar colour and shape of the marker. In legend flux correspond to the flux of star from original image.

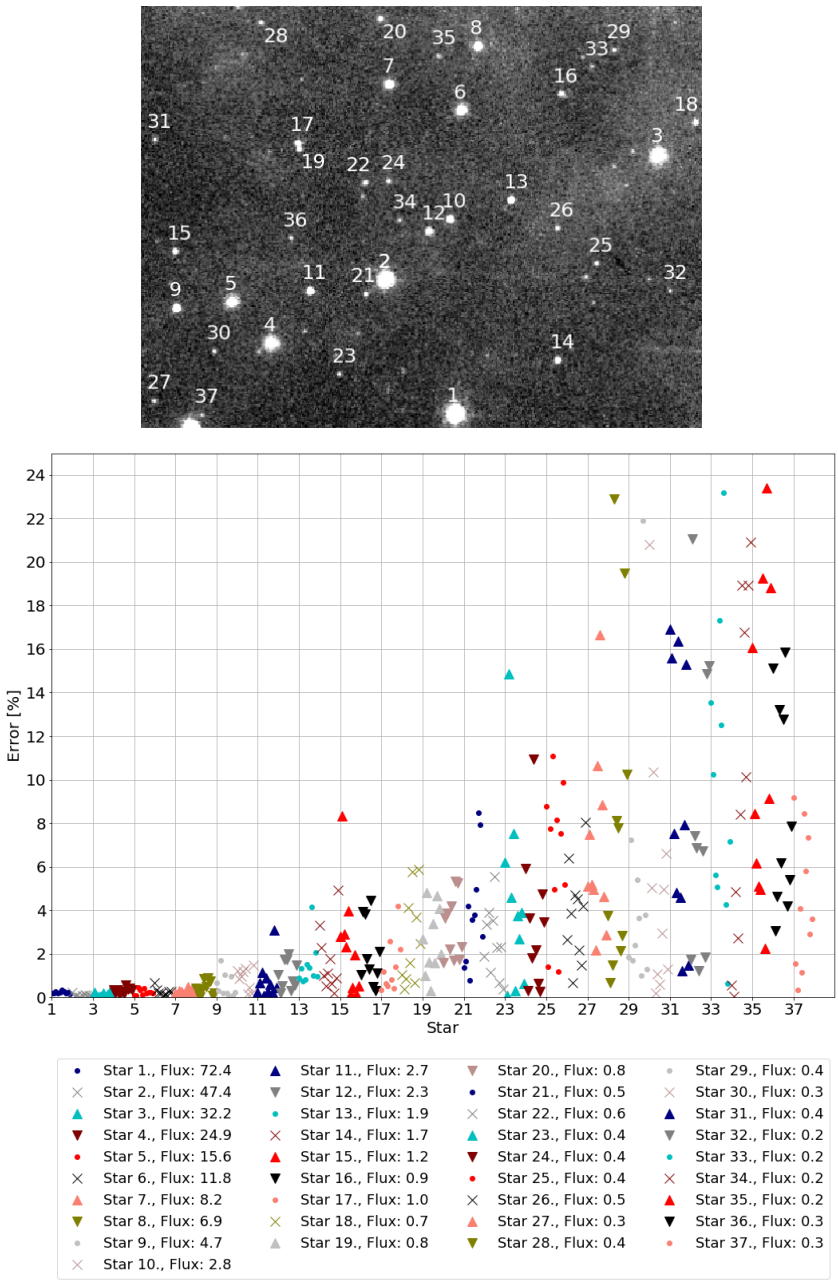

Figure 4.14: Munipack star detection. In comparison with Figure 4.12 we see, that Munipack detected more stars than Star Extractor. The Star Extractor is struggling to distinguish between stars 17 and 19 (star 15 in Figure 4.12) and it has higher error. All of detected stars are within the estimated error of Munipack.

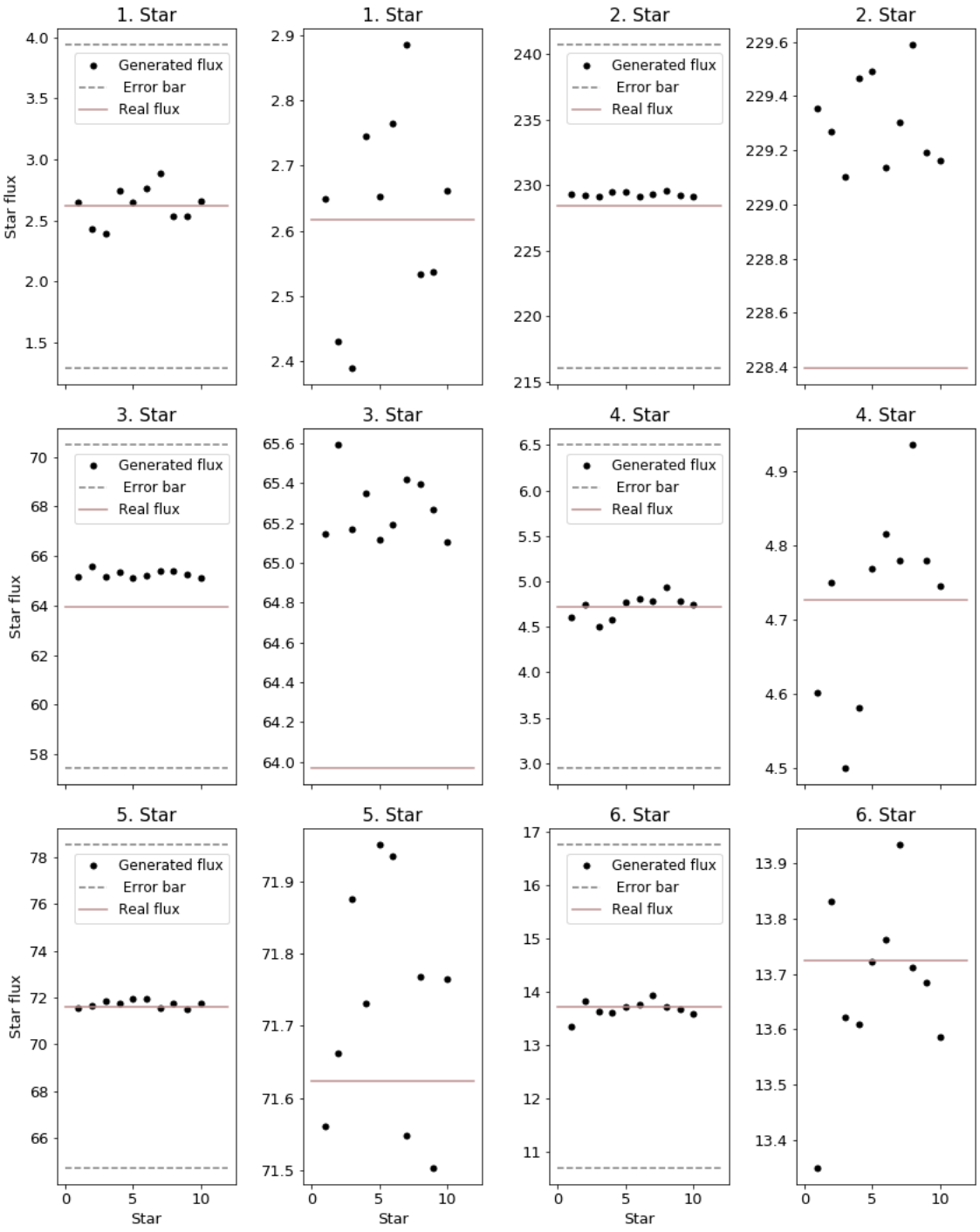

Figure 4.15: Star flux of 6 stars estimated by Star Extractor. Stars correspond to first six stars in Figure 4.13.

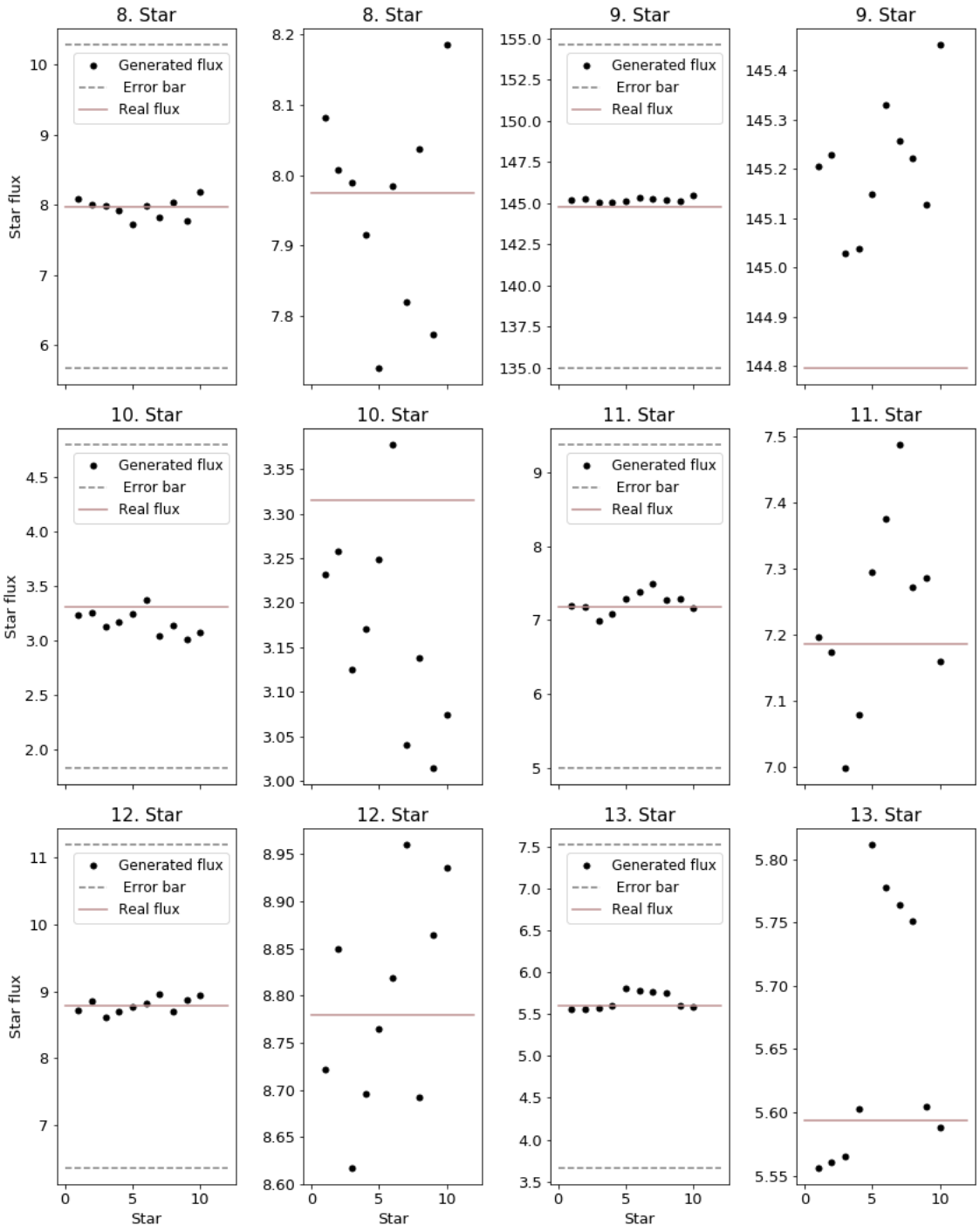

Figure 4.16: Star flux of 6 stars estimated by Star Extractor. Stars correspond to 8. − 13. star in Figure 4.13.

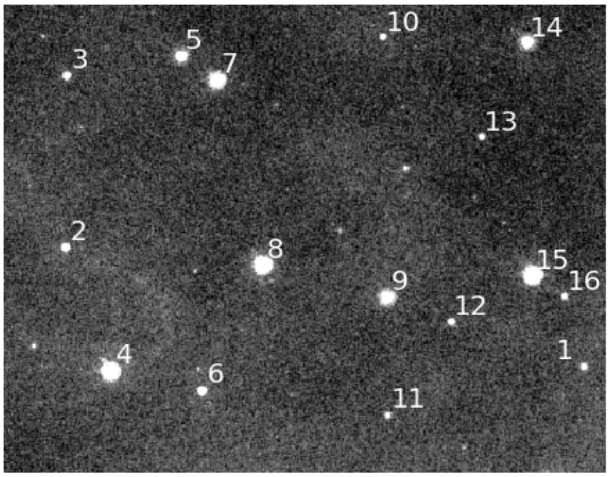

Figure 4.17: Sub-window image contains sixteen stars. The star number is associated with star fluxes in Figure 4.13

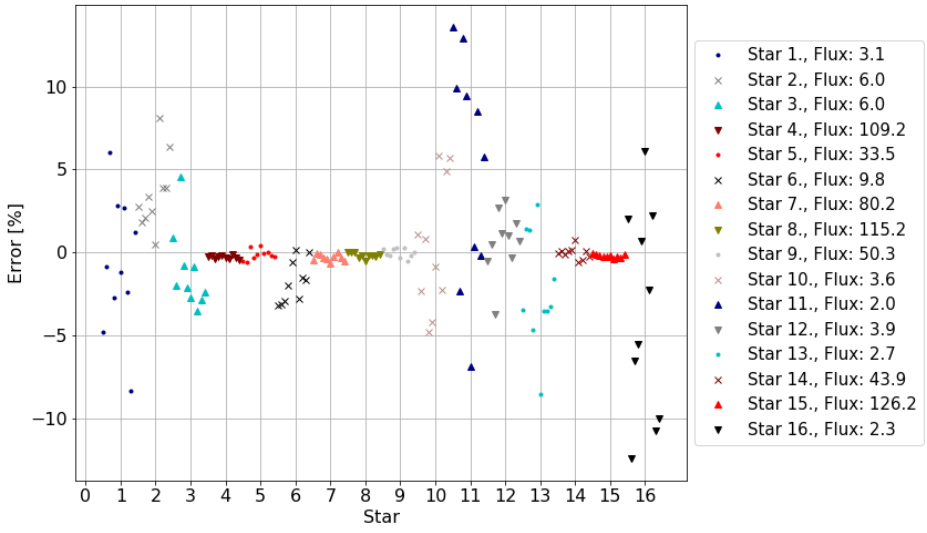

Figure 4.18: The star flux error of sixteen stars from ten times simulated image Figure 4.17. The star error was calculated according Equation 2.32. Same stars have similar colour and shape of the marker. In legend flux correspond to the flux of star from original image.

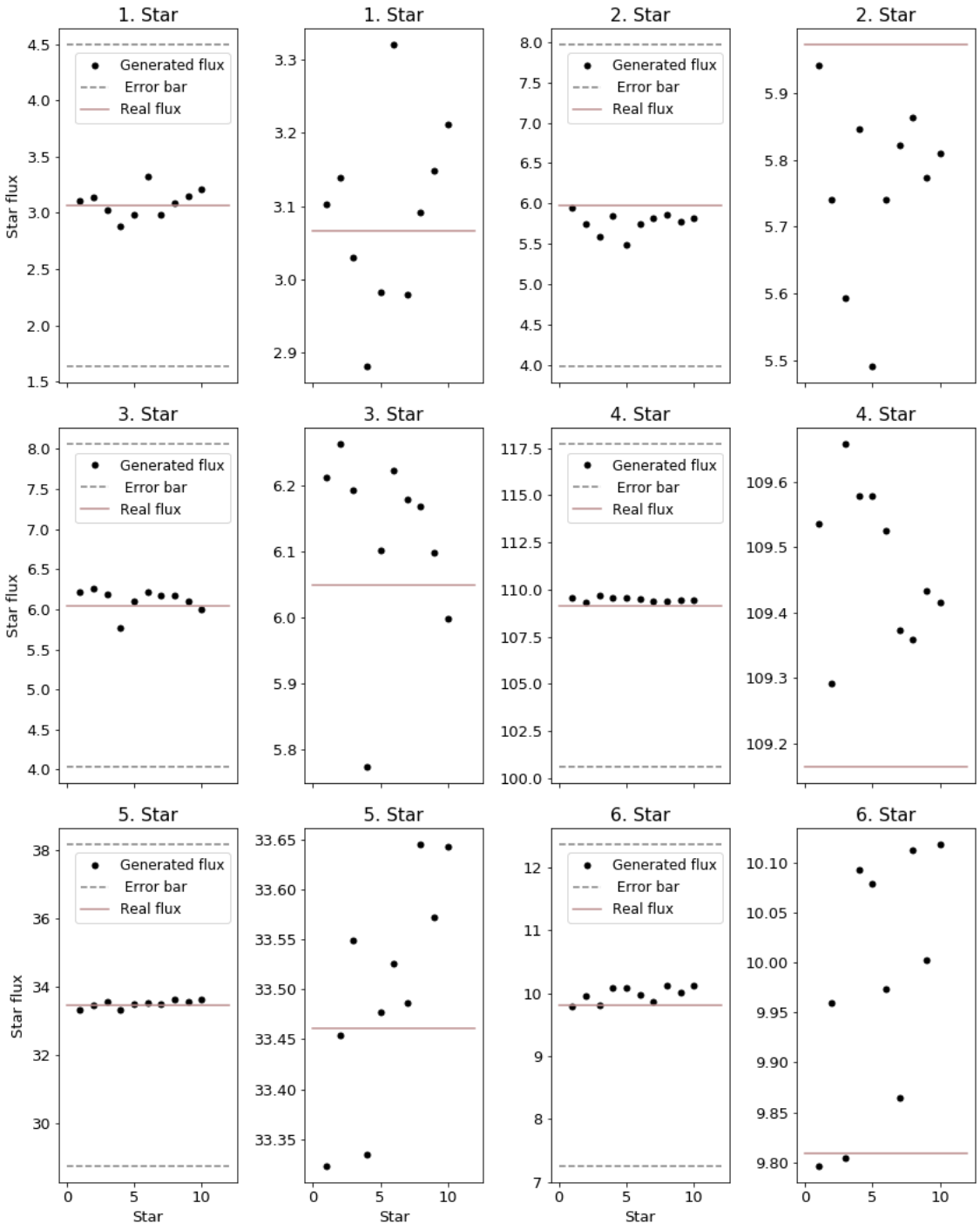

Figure 4.19: Star flux of 6 stars estimated by Star Extractor. Stars correspond to first six stars in Figure 4.18.

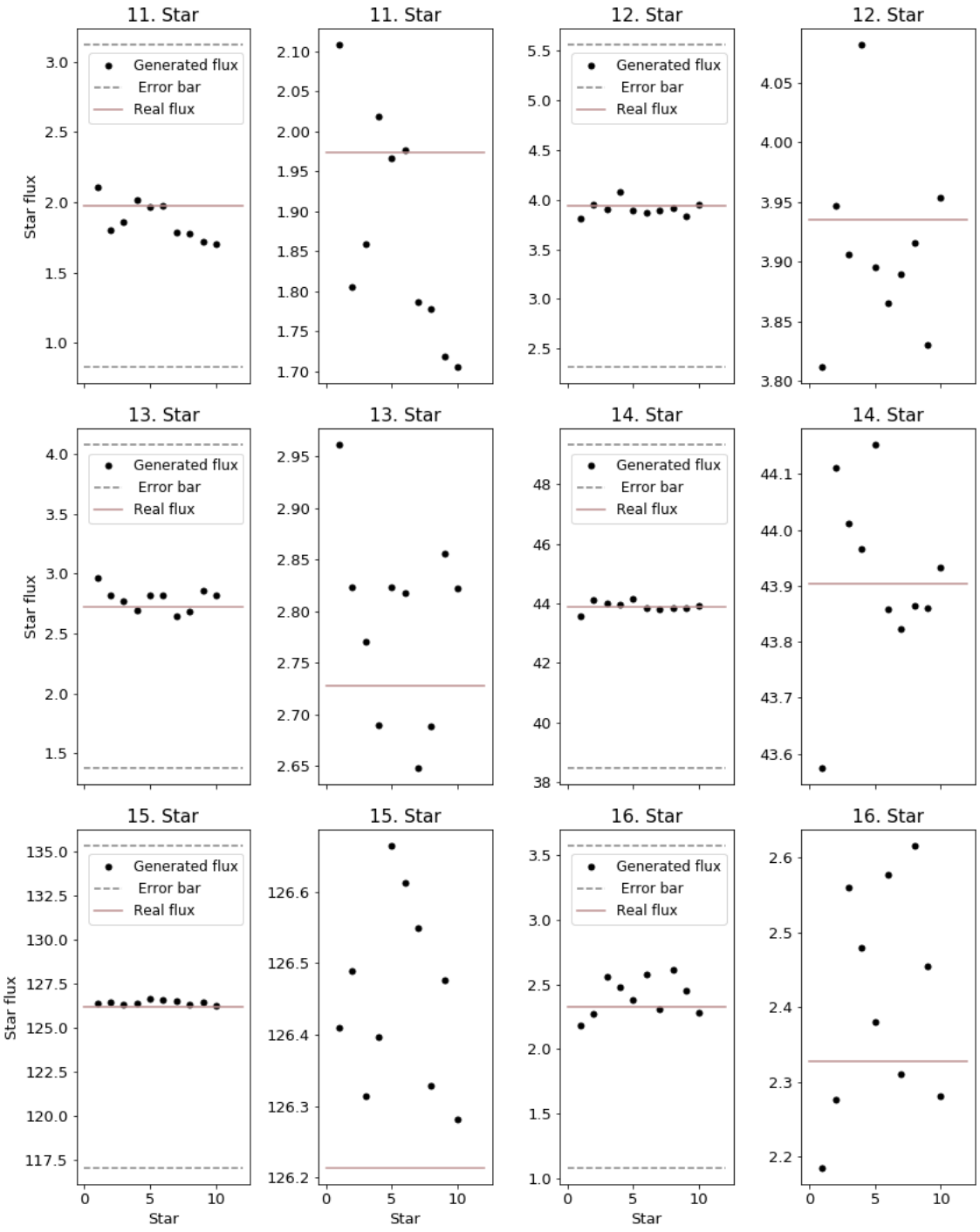

Figure 4.20: Star flux of 6 stars estimated by Star Extractor. Stars correspond to 11. − 16. star in Figure 4.18.

4.2.3 The intensity Distribution

The distribution is another important feature which we want to examined to evaluate the network. As said in the Press et al. [2007]: *"Can we disprove, to a certain required level of significance, the null hypothesis that two data sets are drawn from the same population distribution function?"* In our case, we will try to prove the null hypothesis that our samples are *consistent* with a single distribution function. For this, we choose to apply three statistical tests – Student's t-test, Kolmogorov–Smirnov (KS) test and Kullback–Leibler (KL) divergence. To addition to the test, the histograms (Figure 4.21) and the cumulative distribution functions (CDF, Figure 4.22, Figure 4.26, Figure 4.28...) are plot. We use two approaches to analyse the images:

- Statistic on big images with mixed sources and background
- Statistic on small sub-windows

Statistic for whole images

The null hypothesis of the t-test is that two distributions have the same mean. The mean t-test statistic for the generated images is -0.91, and the mean p-value is 0.49 and for the noisy images -181.82 and 0 respectively. The null hypothesis in case of noisy images can be disproved, but in the case of the generated image, we proved the null hypothesis. The result shows that the mean value of the generated images is closer to the mean of the original images, which can also be seen in Table 4.10.

The KS-test is used to find out if two data samples come from the same distribution. The KS-statistic is done on the normalised images:

$$I_{\text{norm}} = \frac{I - \bar{I}}{\sigma}, \tag{4.1}$$

where I denotes image $\bar{I}$ image mean and σ its standard deviation. In Table 4.8 and Table 4.9 we can see that mean KS-statistic is smaller for the generated images 0.10 than for the images with noise 0.20, which suggest that the cumulative distributions function of the generated and original images are more similar.

The experiments with CDF show the following trend: more noise we add to the image the slope of the CDF line decrease, the CDF is wider and more shifted to the higher values. This is also seen in Figure 4.22 and on the histograms shown in Figure 4.21 which follows same trend. The images with noise are always shifted and more extended in comparison with the original

Image	T-test statistic	T-test p-value	KS-test statistic	KL divergence 10^{-7}
1	-0.36	0.72	0.02	9.93
2	-0.62	0.54	0.03	3.38
3	-1.97	0.05	0.12	1.91
4	-0.93	0.35	0.15	-9.18
5	-2.64	0.01	0.12	2.92
6	0.04	0.97	0.16	2.53
7	-0.16	0.88	0.15	7.80
8	-0.90	0.37	0.11	2.02
9	-0.61	0.54	0.10	5.95
Mean	-0.91	0.49	0.10	3.03

Table 4.8: The summary of the t-test, KS-test and KL divergence for the generated images. The p-value of the KS-test is not shown because despite the statistic is close to zero, the p-value was zero for all images.

images. Conversely, the CDF of the generated images shows that slope of the line and the value range is in good match with the original one which signifies the noise reduction. The Figure 4.23 shows normalised CDF and it's residuals. We observe that the residuals of the generated images are close to zero in the range of $0.2 - 0.8$, that implies it is a good fit.

The KL-divergence reveals that approximate the original distribution by generated image cost less loss of information 3.03 than approximation by the noisy image 11.21. Nevertheless, we can see that with the generated image, there is some loss of information.

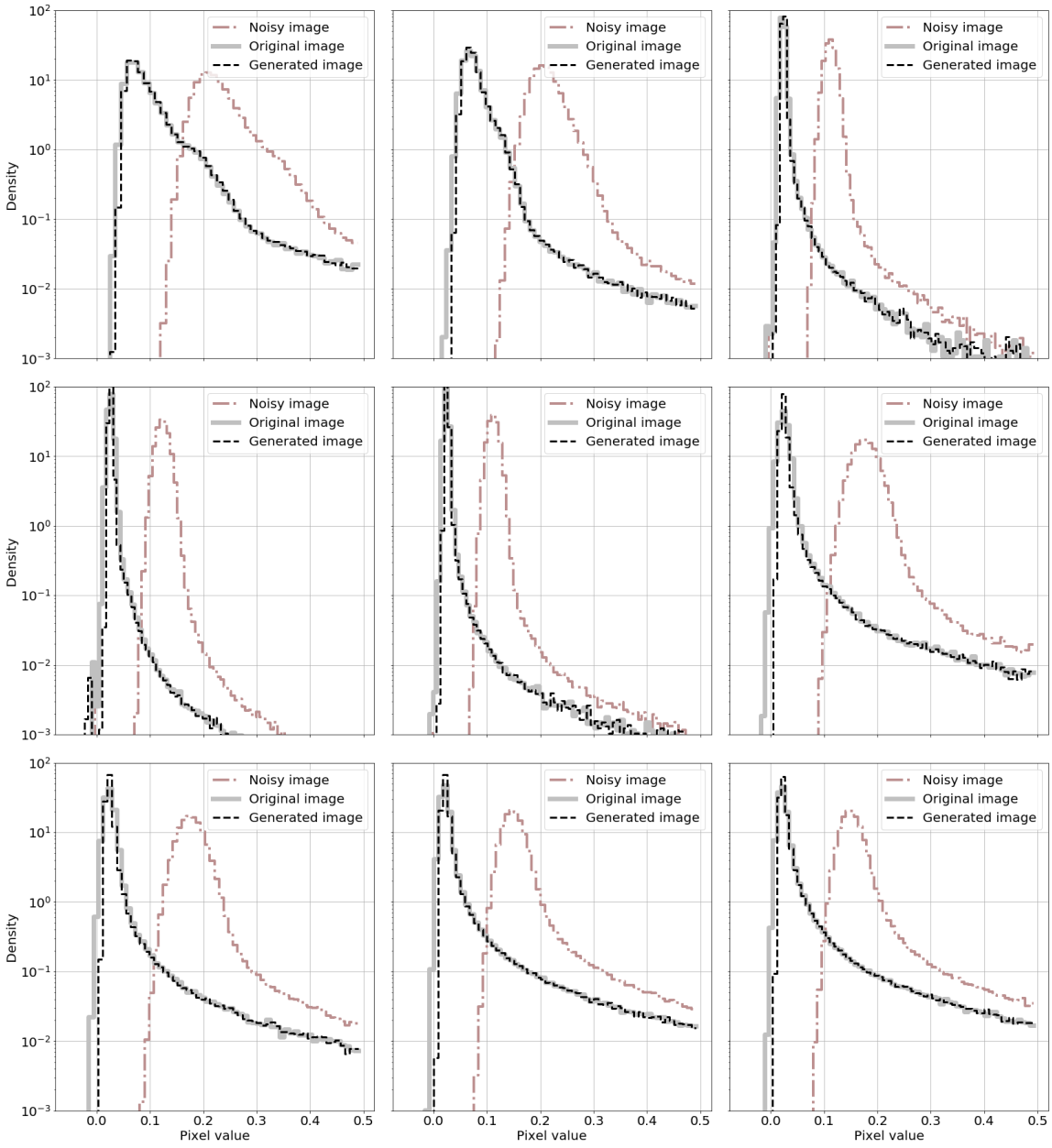

Figure 4.21: Example of the distribution for the six first images from Table 4.8 and 4.9

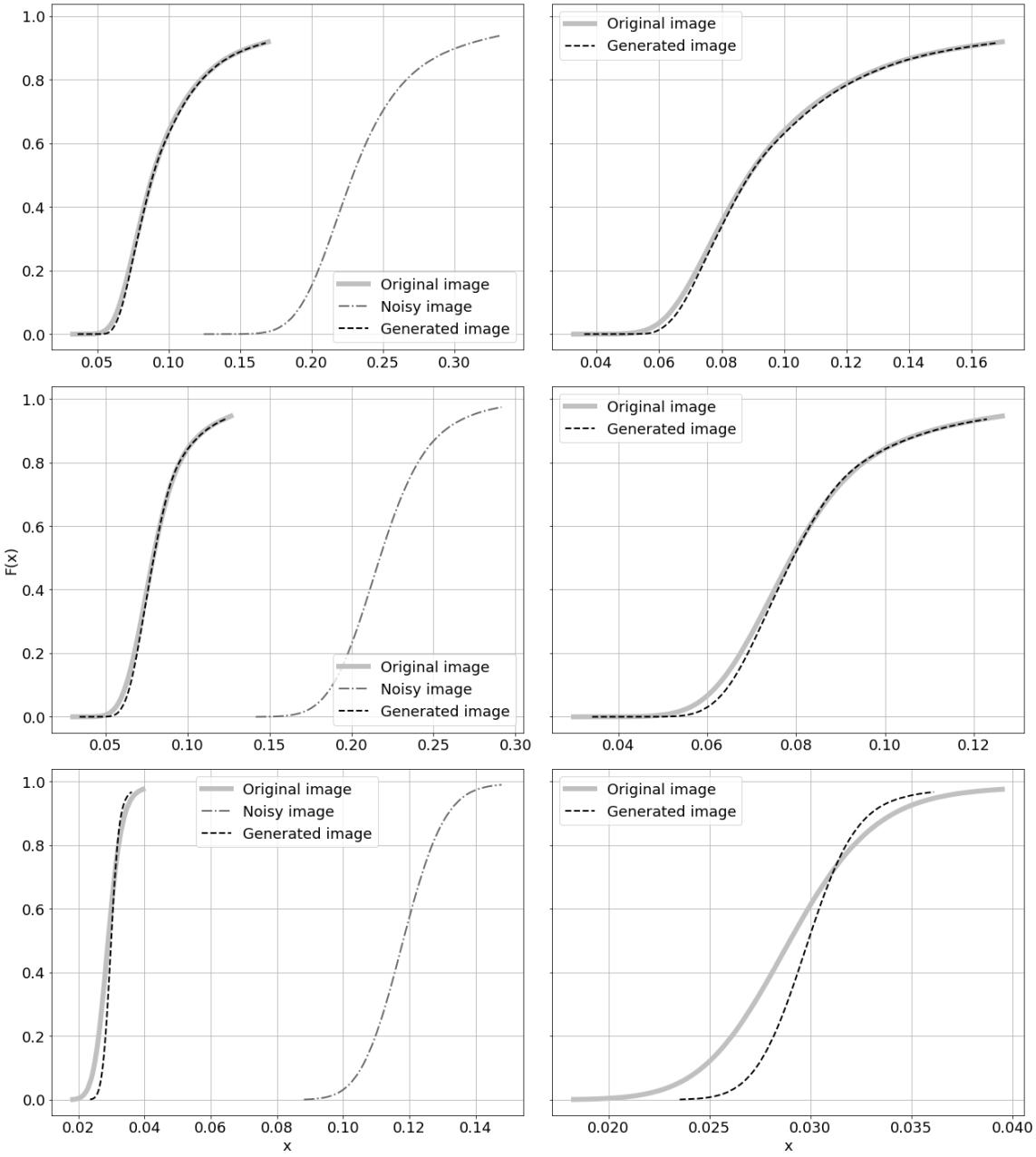

Figure 4.22: Cumulative distribution function for first three images from Table 4.8 and 4.9 and Figure 4.21. The figures the left correspond to CDF of the original, noisy and generated image and the right ones are CDF of the original and generated images.

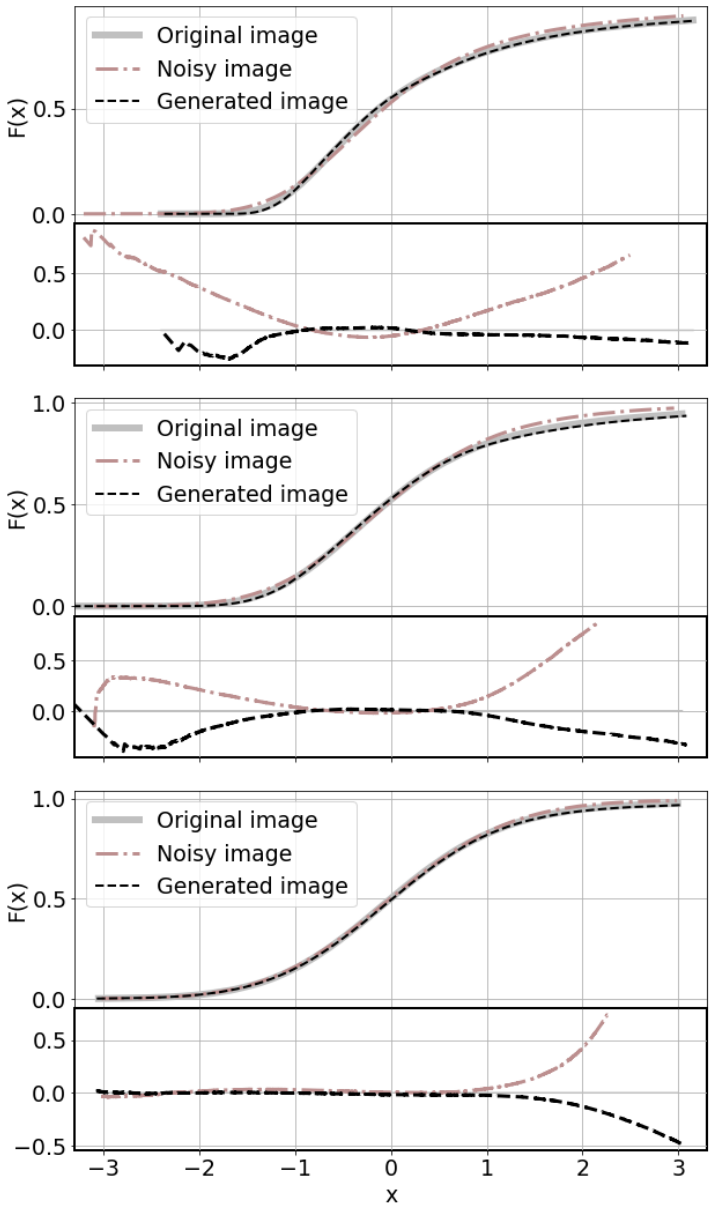

Figure 4.23: The left figure shows normalised CDF and right figure the residual of the of the CDF from Figure 4.22. The CDF correspond to the first three images from Table 4.8 (Generated images), 4.9 (Noisy images) and Figure 4.21.

Image	T-test statistic	T-test p-value	KS-Test statistic	KL divergence 10^{-7}
1	-40.76	0.0	0.10	15.88
2	-88.19	0.0	0.13	12.13
3	-192.54	0.0	0.24	4.37
4	-625.25	0.0	0.22	-4.78
5	-217.47	0.0	0.24	5.60
6	-183.07	0.0	0.22	19.38
7	-103.23	0.0	0.22	22.63
8	-114.3	0.0	0.21	11.84
9	-71.55	0.0	0.21	13.84
Mean	-181.82	0.0	0.20	11.21

Table 4.9: The summary of the t-test, KS-test and KL divergence for the noisy images. The p-value of the KS-test is not shown because despite the statistic is close to zero; the p-value was zero for all images.

	Mean			Median		
Image	Original	Generated	Noise	Original	Generated	Noise
1	0.1522	0.1534	0.2910	0.0887	0.0890	0.2291
2	0.1020	0.1030	0.2394	0.0789	0.0793	0.2173
3	0.0322	0.0331	0.1213	0.0290	0.0299	0.1181
4	0.0335	0.0336	0.1313	0.0322	0.0324	0.1300
5	0.0327	0.0337	0.1215	0.0296	0.0307	0.1185
6	0.0482	0.0482	0.2018	0.0316	0.0310	0.1852
7	0.0564	0.0566	0.2102	0.0320	0.0315	0.1858
8	0.0641	0.0651	0.1929	0.0287	0.0290	0.1580
9	0.0734	0.0745	0.2025	0.0294	0.0297	0.1590

Table 4.10: The summary of mean, median and standard deviation of the original, generated and noisy images.

Statistic for image sub-windows

In Table 4.11 ten different areas from image Figure 4.24 are studied – five containing source and five pure background areas. As it is obvious from Table 4.11 that the T-statistic is in every examined area better for generated images than for the noisy ones (Table 4.12). This indicates that mean of the generated images is close to the original images. On the other hand, we notice the background areas have significantly poorer results. The generated images have roughly 4 times lower KL-Divergence than the noisy ones; the KS-statistic is approximately similar in both cases.

The pixel distribution in Figure 4.25 confirms that the generated images fit the original better than the noisy images. However, there is still space for improvement. From the CDF of sub-windows containing source shown in Figure 4.26, we observed that the network could reconstruct the mean value of the original image well. Nevertheless, the CDFs of the generated images are comparatively narrower than the original ones. This mean, that noise is in generated image reduced, images are less noisy than the original images. The Figure 4.27 show the normalised CDF from previously mentioned plot and the residuals. From residual, we observe that there are minimal variations between the original and generated image.

The Figure 4.28 shows the CDFs for the sub-window of the background. It is evident, that background sub-windows across the original image shares similar features; they have almost similar CDFs. The same goes for the sub-windows from the generated image, although the shifts of mean values are visible and the slope of the line is higher, which indicates the noise reduction. After normalisation, shown in Figure 4.29 we see, that the generated data fits the original well, the residuals show small variation around zero.

The Figure 4.30 display second examined image of the galaxy NGC 3344. In the Table 4.13 one can find the statistical test results for ten sub-windows from generated image.

A mean KL-divergence for sub-windows from the generated images is 5.2, which is lower than for the noisy images with a mean KL-divergence 12.3. A mean KS-statistic for the generated image is 0.03; for the noisy images, the result is 0.06. The pixel distribution of the images is shown in Figure 4.31.

The T-statistic reveals the same trend as in the previous example – the mean values of the generated images are closer to original images. In the histograms shown in Figure 4.31, we see that the distribution of the images with noise is wider and shifted to the left. The distributions of the generated images are shifted closer to the original image distribution, which underlines the correctness of our method. The Figure 4.32 and Figure 4.34 shows CDFs

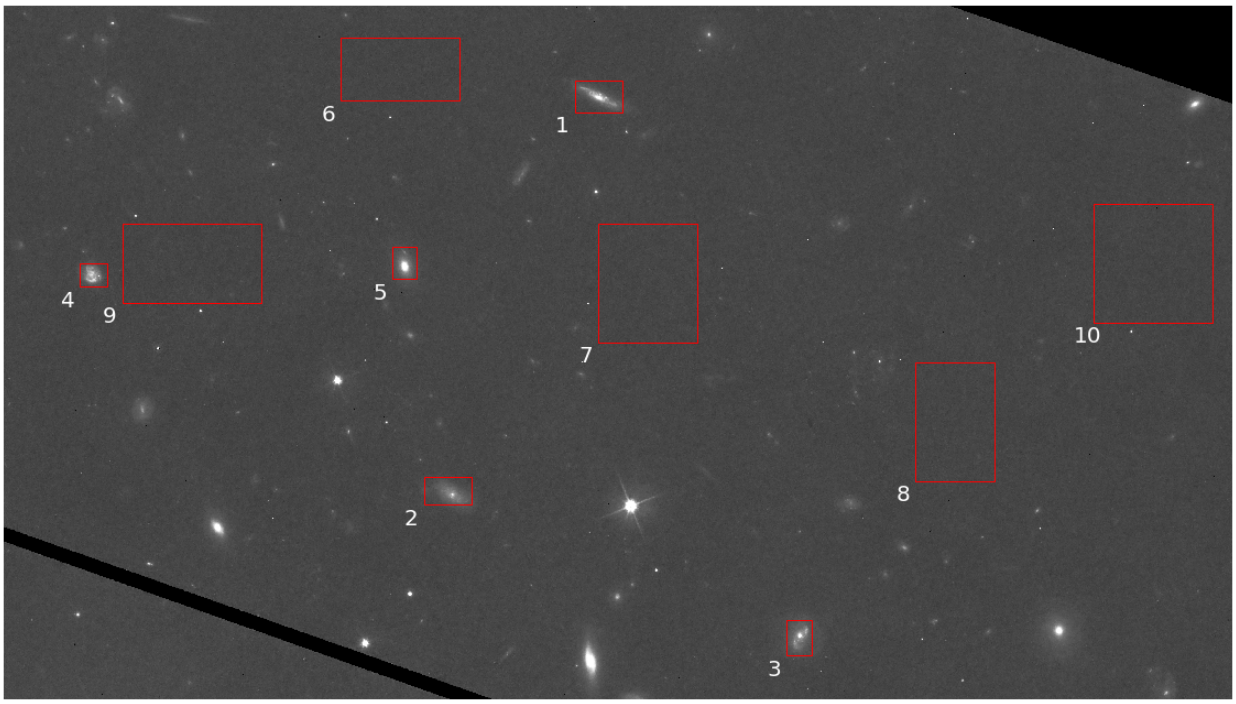

Figure 4.24: Red shows rectangles are examined areas for Table 4.11 and Table 4.12. The first five areas contains different sources and another five areas the background.

of sub-windows which have different type and density of stars and the dust of the galaxy. The CDFs of the generated and original images are more similar than the CDFs of the noisy and original images. When we study the corresponding normalised CDF and their residuals (Figure 4.33 and Figure 4.35) we see small variations close to zero in $F(x)$ range $0.2 - 0.8$.

Tests done on the sub-windows indicates that distributions of the generated images are more similar to the original images than the distributions of the noisy images, although they are not the same. The problem can be partially caused by simulated noise added to the images.

Image	T-test statistic	T-test p-value	KS-Test statistic	KL divergence 10^{-7}
1	-0.36	0.72	0.07	0.14
2	2.32	0.02	0.01	0.71
3	0.62	0.54	0.01	1.41
4	0.91	0.36	0.06	0.69
5	0.42	0.67	0.04	1.50
6	-2.66	0.01	0.08	1.05
7	4.08	0.0	0.04	0.97
8	-19.3	0.0	0.03	1.24
9	27.6	0.0	0.04	0.8
10	-28.14	0.0	0.02	1.32

Table 4.11: The summary of the t-test, KS-test and KL divergence for the parts of the generated images on Figure 4.24. The p-value of the KS-test is not shown because despite the statistic is close to zero, the p-value was zero for all images. The first 5 rows corresponds to the different sources and rest to the background.

Image	T-test statistic	T-test p-value	KS-Test statistic	KL divergence 10^{-7}
1	-171.22	0.0	0.09	3.21
2	-270.5	0.0	0.06	3.76
3	-90.1	0.0	0.07	4.4
4	-112.06	0.0	0.05	3.52
5	-55.93	0.0	0.07	4.55
6	-1741.26	0.0	0.01	4.24
7	-2149.47	0.0	0.01	4.1
8	-1931.68	0.0	0.01	4.42
9	-2051.46	0.0	0.02	3.91
10	-2412.72	0.0	0.01	4.55

Table 4.12: The summary of the t-test, KS-test and KL divergence for the parts of the noisy images on Figure 4.24. The p-value of the KS-test is not shown because despite the statistic is close to zero, the p-value was zero for all images. The first 5 rows corresponds to the different sources and rest to the background.

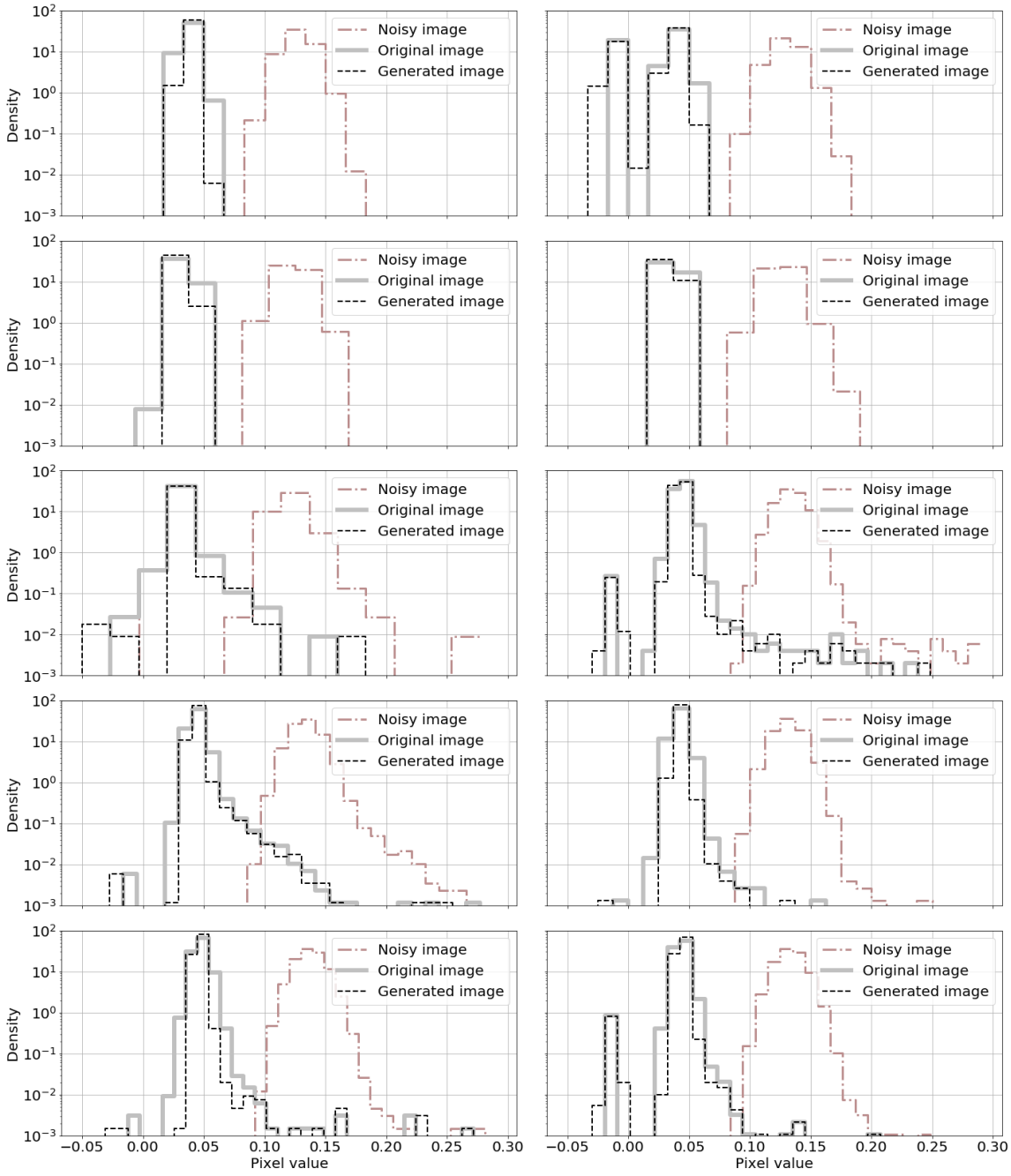

Figure 4.25: The distribution of ten sub-windows from image Figure 4.24. The statistic is shown in Table 4.11 and 4.12

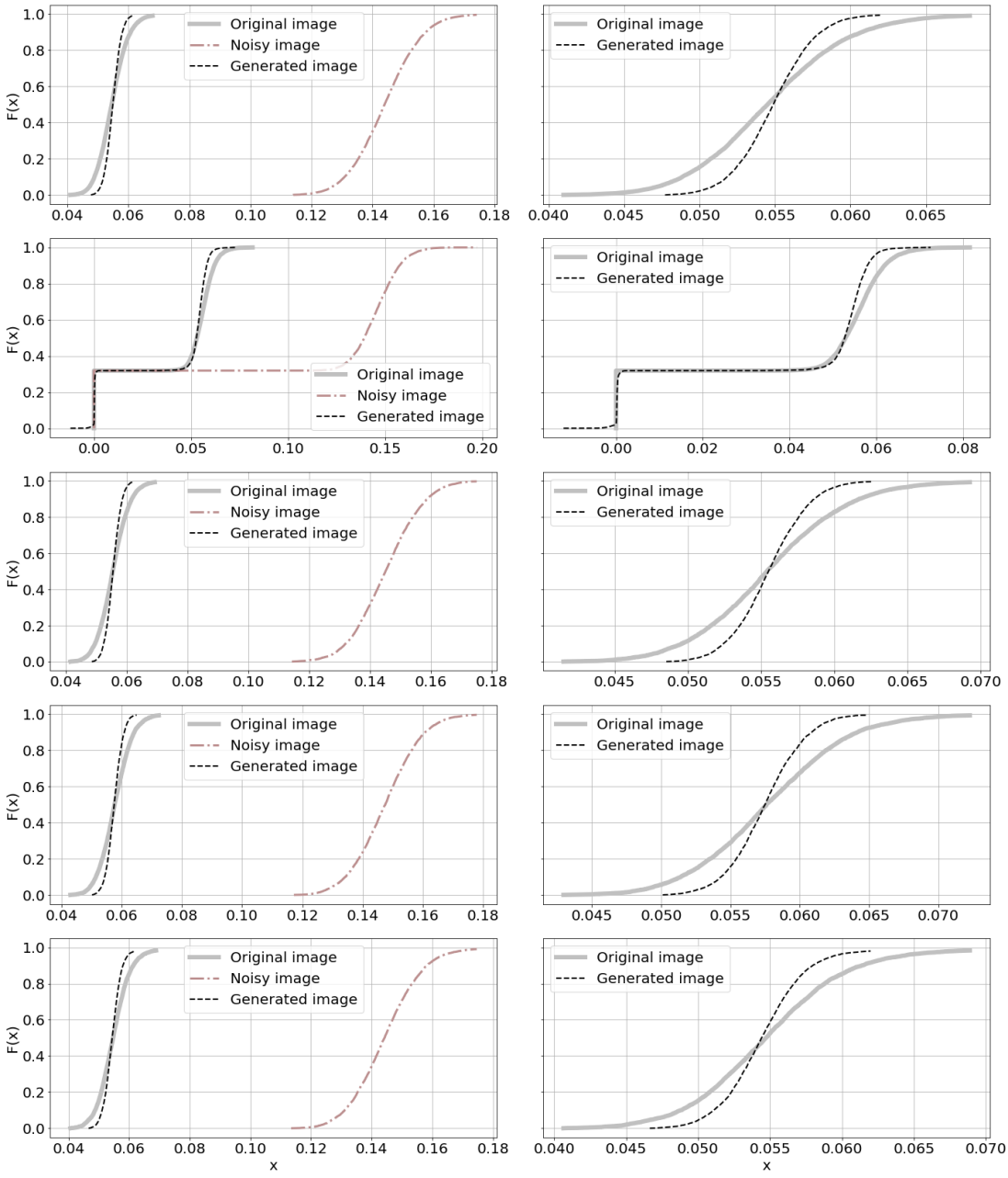

Figure 4.26: The cumulative distribution function of first five sub-windows (1-5) from image Figure 4.24 and first five histograms shown in Figure 4.25. The statistic is shown in Table 4.11 and 4.12

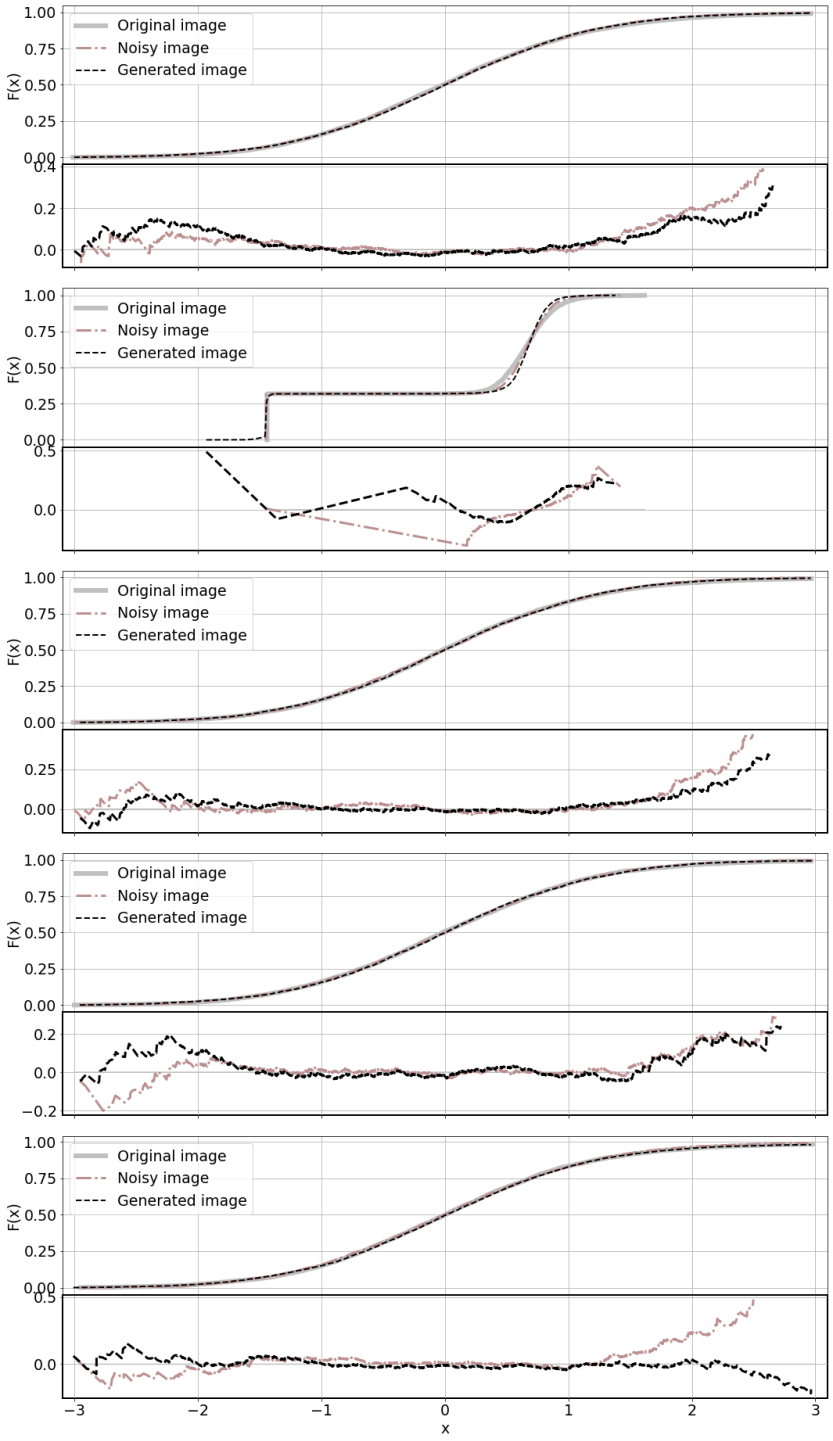

Figure 4.27: The left figure shows normalised CDF and right figure the residual of the of five sub-windows from Figure 4.26

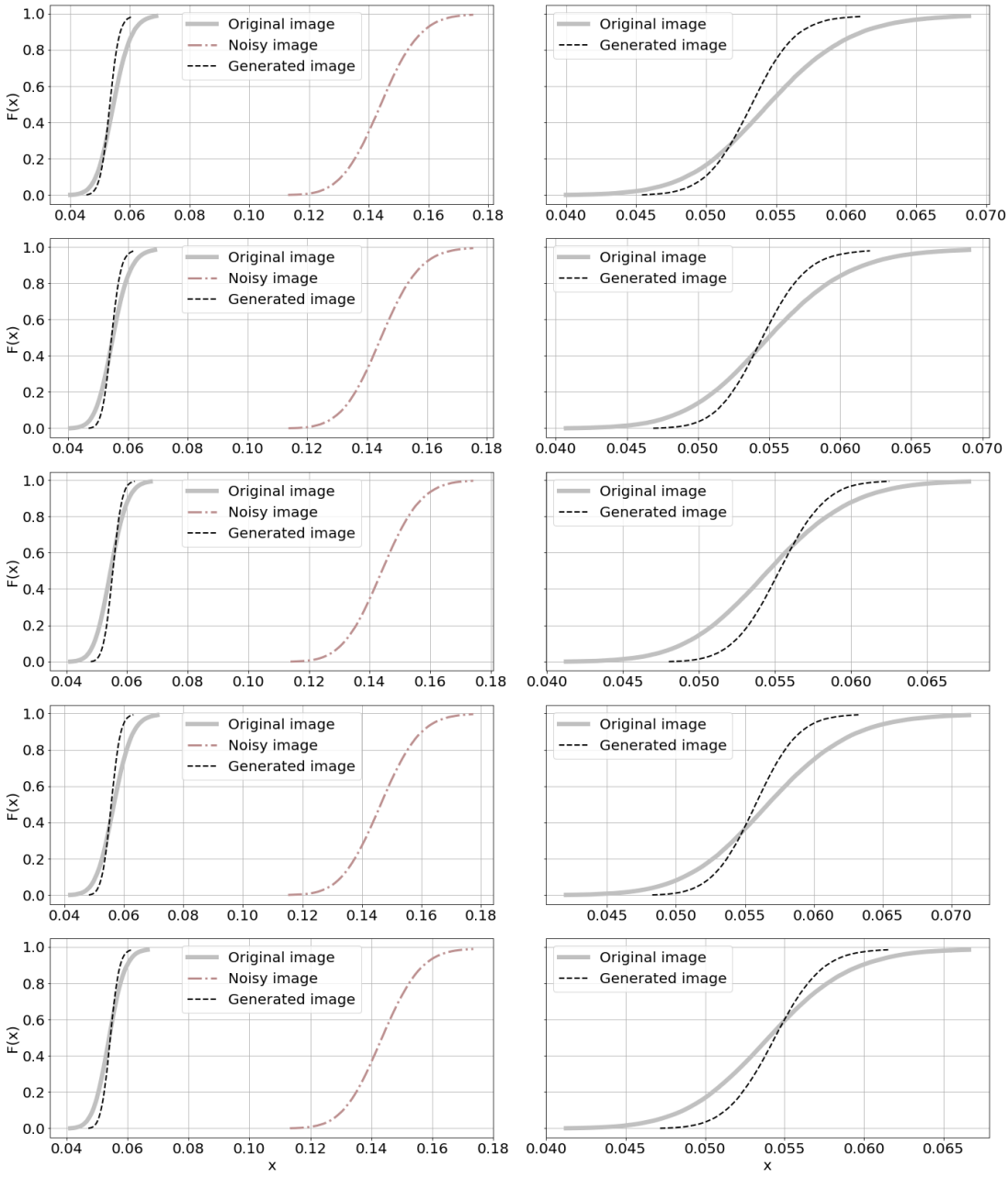

Figure 4.28: The cumulative distribution function of second five sub-windows (6-10) from image Figure 4.24 and second five histograms shown in Figure 4.25. The statistic is shown in Table 4.11 and 4.12

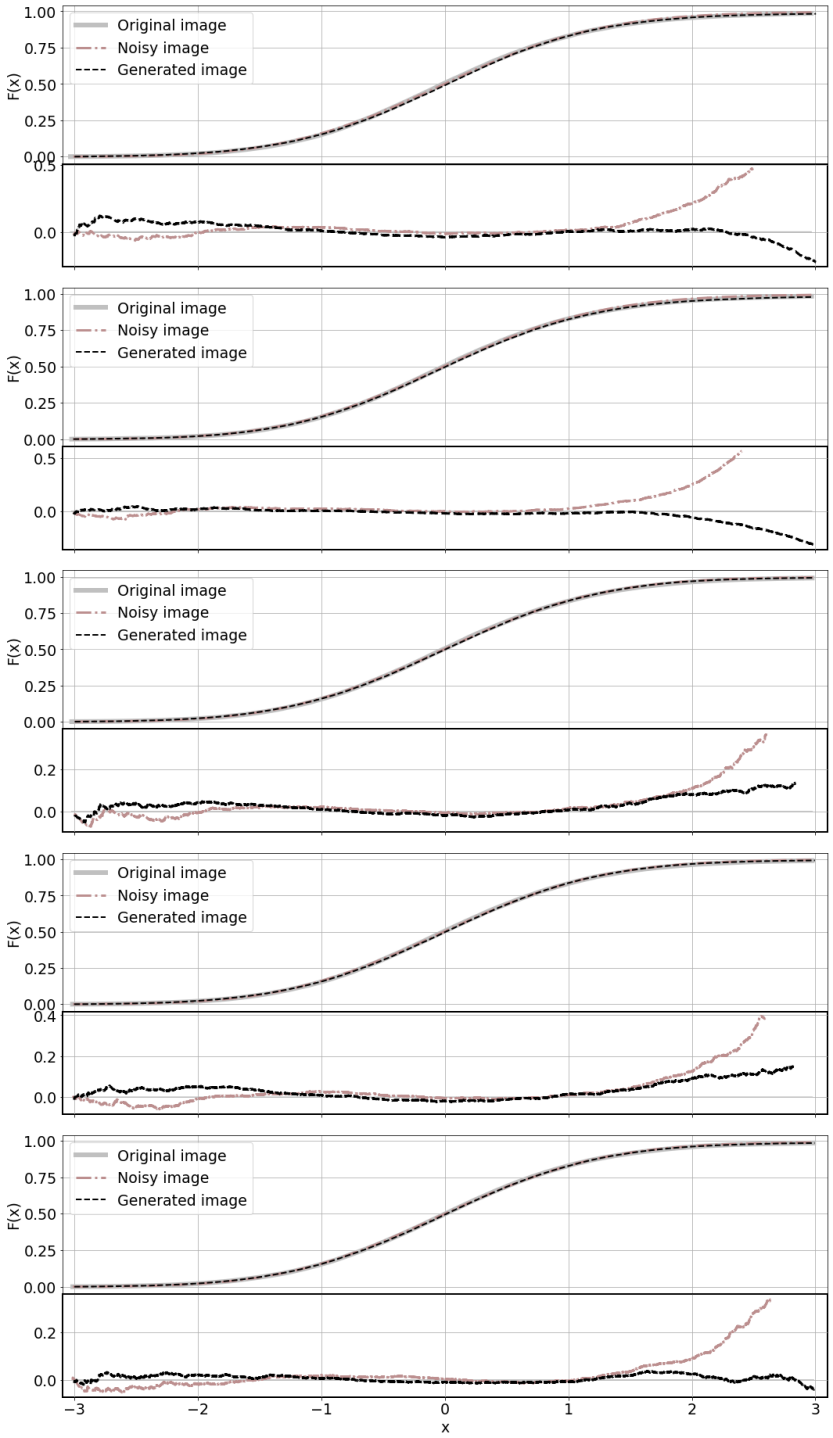

Figure 4.29: The left figure shows normalised CDF and right figure the residual of the of five sub-windows from Figure 4.28

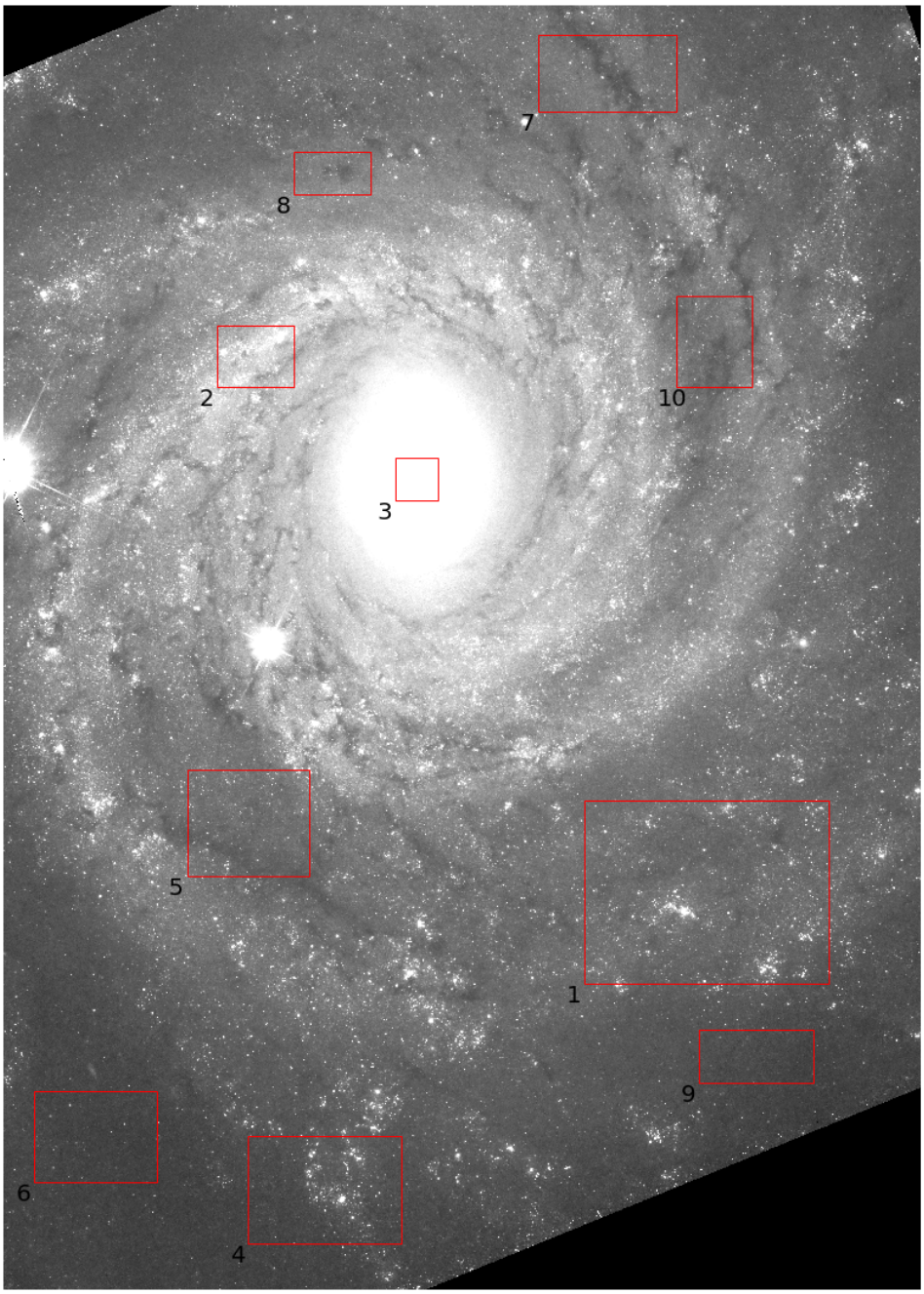

Figure 4.30: Red shows rectangles are examined areas for Table 4.13 and Table 4.14

Image	T-test statistic	T-test p-value	KS-Test statistic	KL divergence 10^{-7}
1	-0.3	0.76	0.05	2.13
2	-0.73	0.46	0.04	3.85
3	-0.07	0.94	0.05	25.16
4	-1.95	0.05	0.03	1.8
5	18.98	0.0	0.04	4.05
6	4.85	0.0	0.02	2.88
7	-3.26	0.0	0.01	3.86
8	0.12	0.91	0.01	3.31
9	8.8	0.0	0.01	1.84
10	14.64	0.0	0.01	2.62

Table 4.13: The summary of the t-test, KS-test and KL divergence for the parts of the generated images on Figure 4.30. The p-value of the KS-test is not shown because despite the statistic is close to zero, the p-value was zero for all images.

Image	T-test statistic	T-test p-value	KS-Test statistic	KL divergence 10^{-7}
1	-1347.63	0.0	0.10	10.31
2	-184.94	0.0	0.04	11.02
3	-3.27	0.0	0.01	21.75
4	-580.11	0.0	0.13	10.53
5	-1383.44	0.0	0.06	12.02
6	-2185.74	0.0	0.05	11.97
7	-1241.32	0.0	0.06	12.19
8	-858.10	0.0	0.03	11.52
9	-1594.61	0.0	0.04	10.66
10	-1206.12	0.0	0.03	10.57

Table 4.14: The summary of the t-test, KS-test and KL divergence for the parts of the noisy images on Figure 4.30. The p-value of the KS-test is not shown because despite the statistic is close to zero, the p-value was zero for all images.

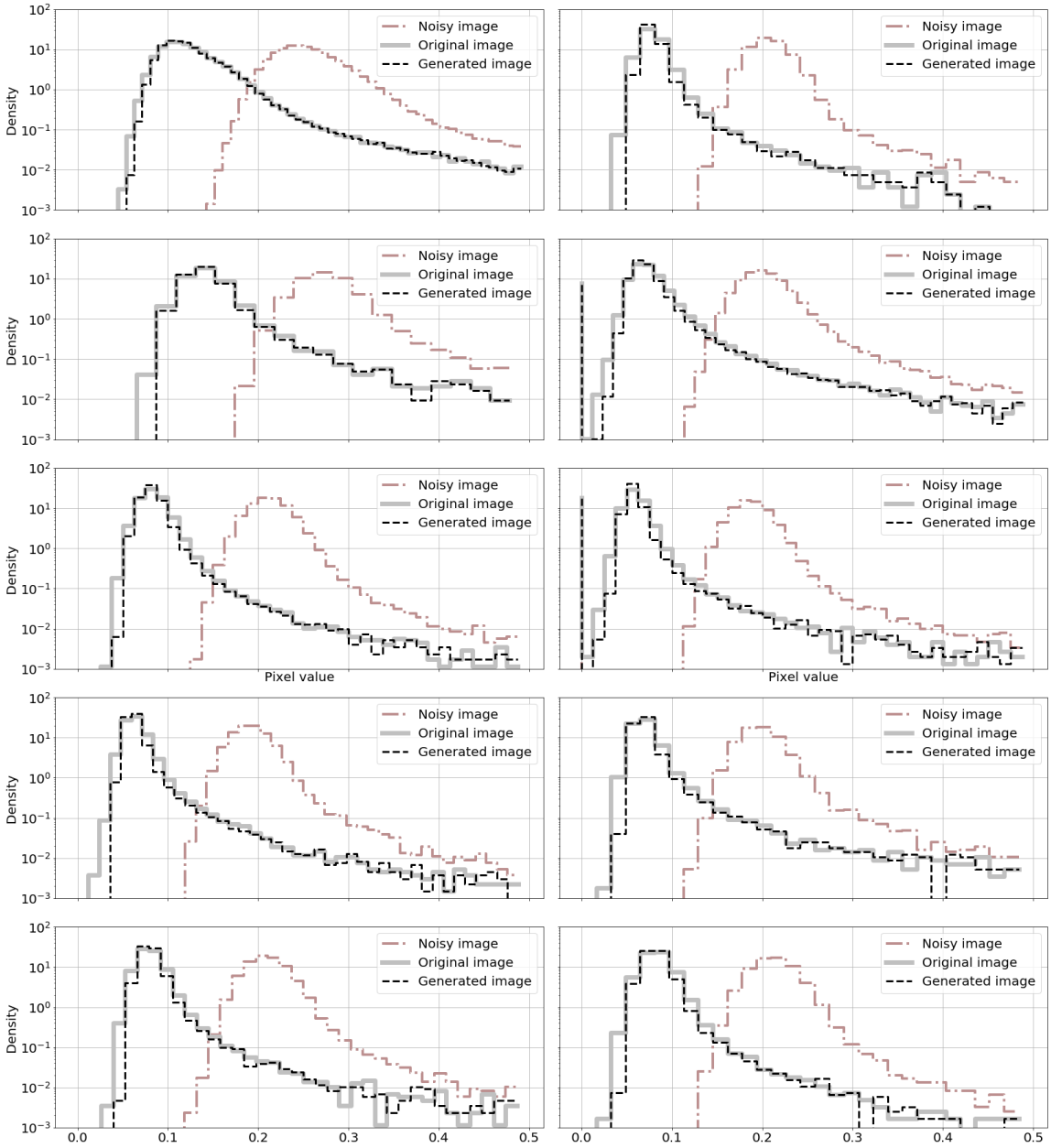

Figure 4.31: The distribution of ten sub-windows from image Figure 4.30. The statistic is shown in Table 4.13 and 4.14

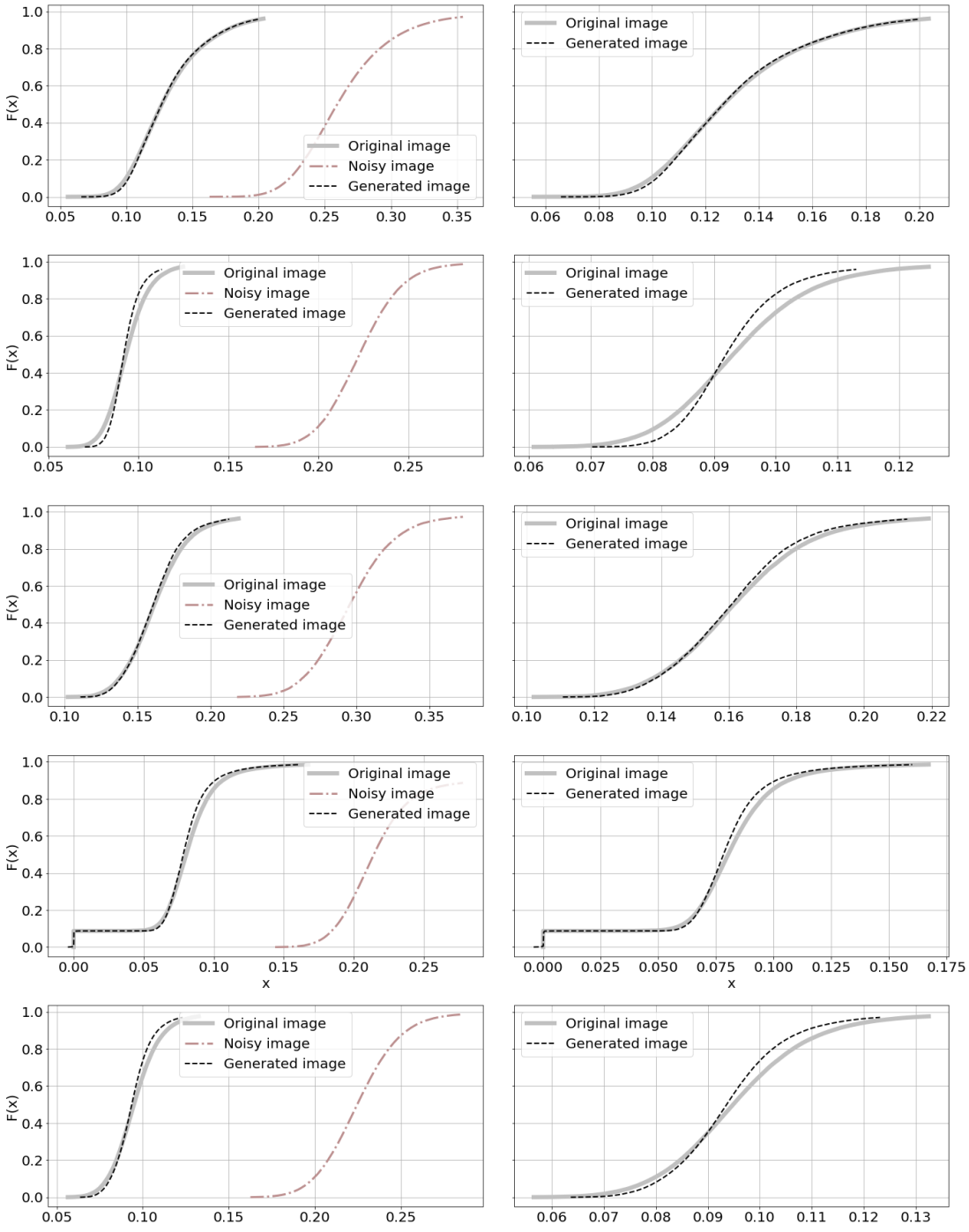

Figure 4.32: The cumulative distribution function of first five sub-windows (1-5) from image Figure 4.30 and first five histograms shown in Figure 4.31. The statistic is shown in Table 4.13 and 4.14

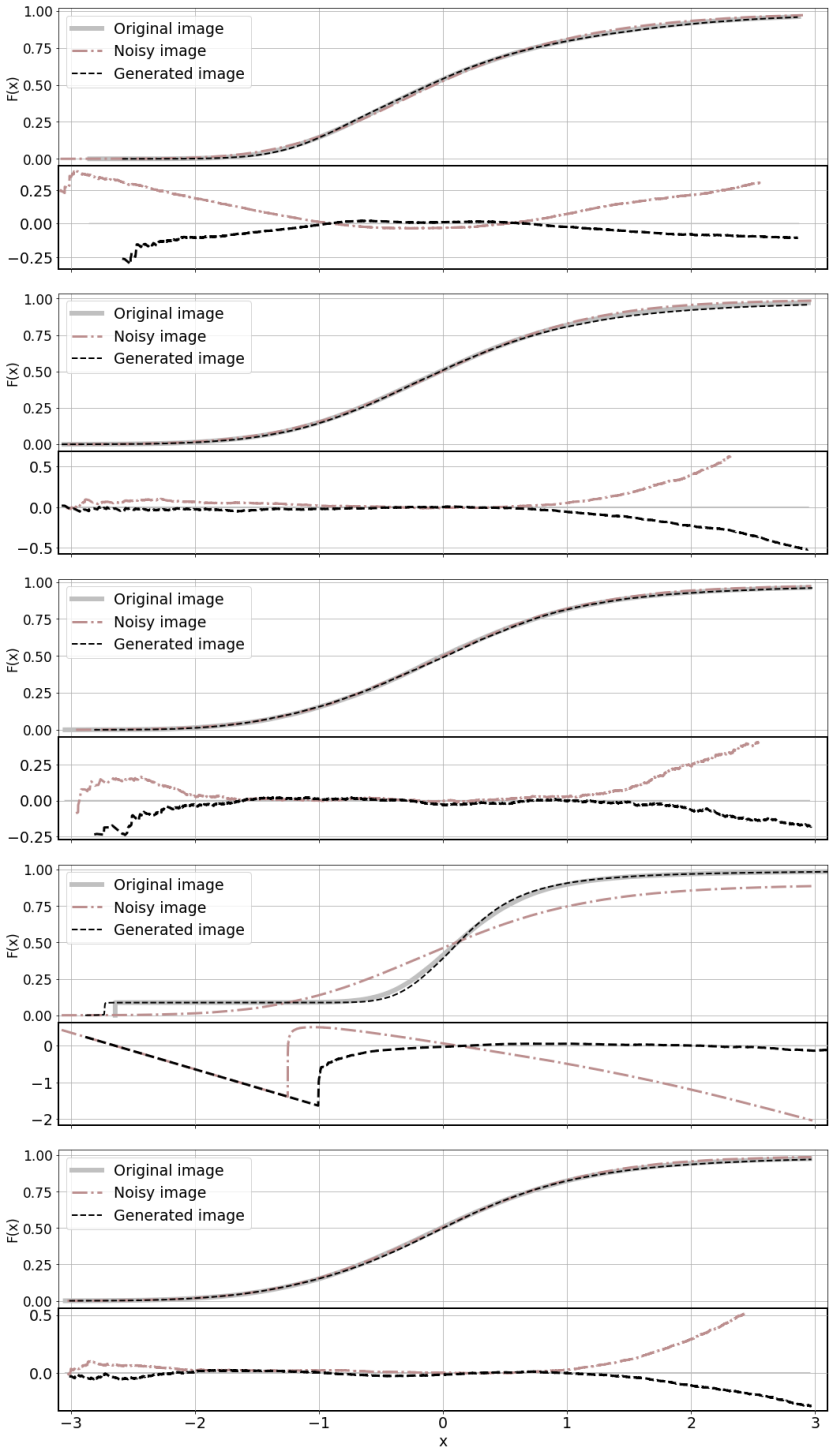

Figure 4.33: The left figure shows normalised CDF and right figure the residual of the of five sub-windows from Figure 4.32

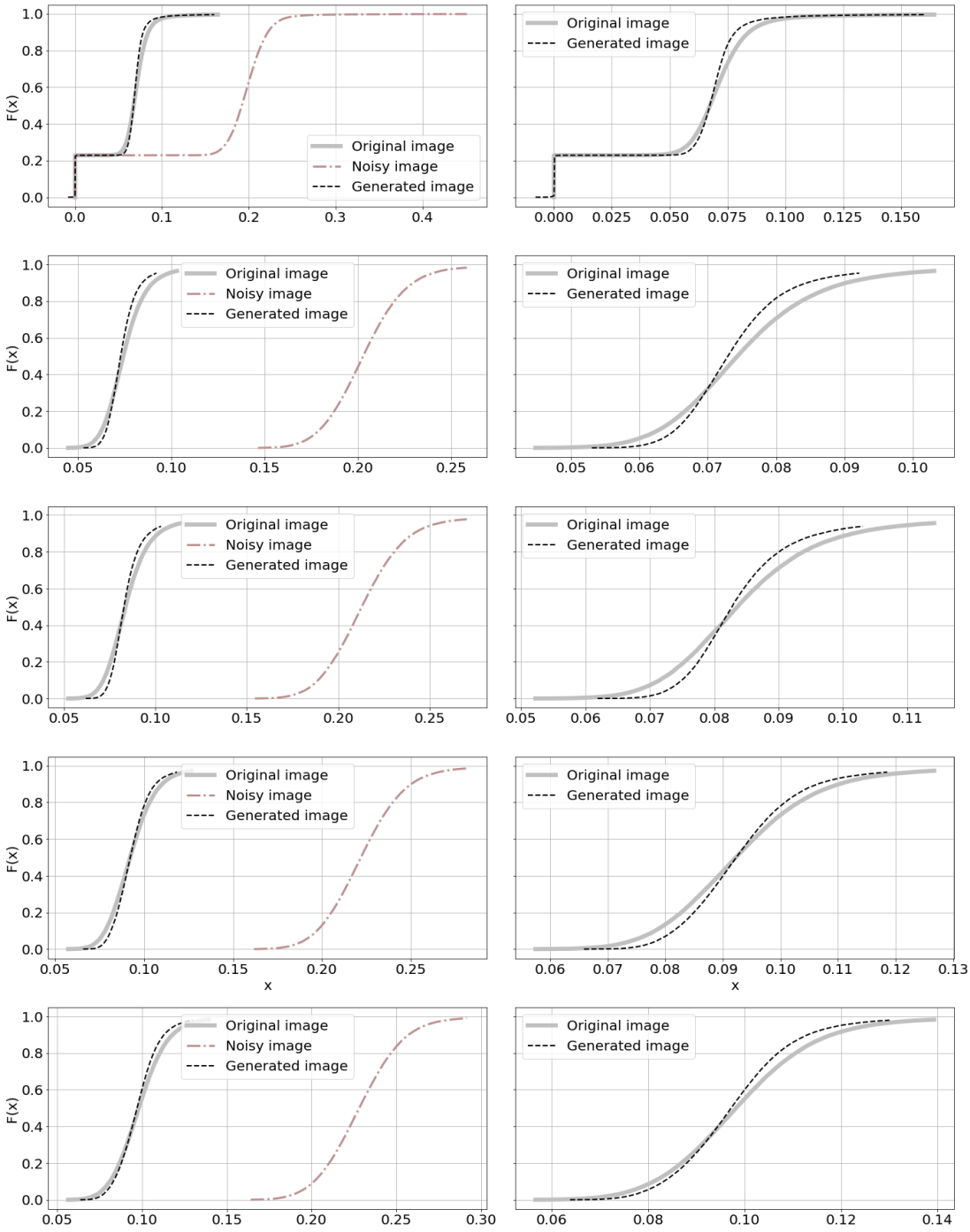

Figure 4.34: The cumulative distribution function of second five sub-windows (6-10) from image Figure 4.30 and second five histograms shown in Figure 4.31. The statistic is shown in Table 4.13 and 4.14

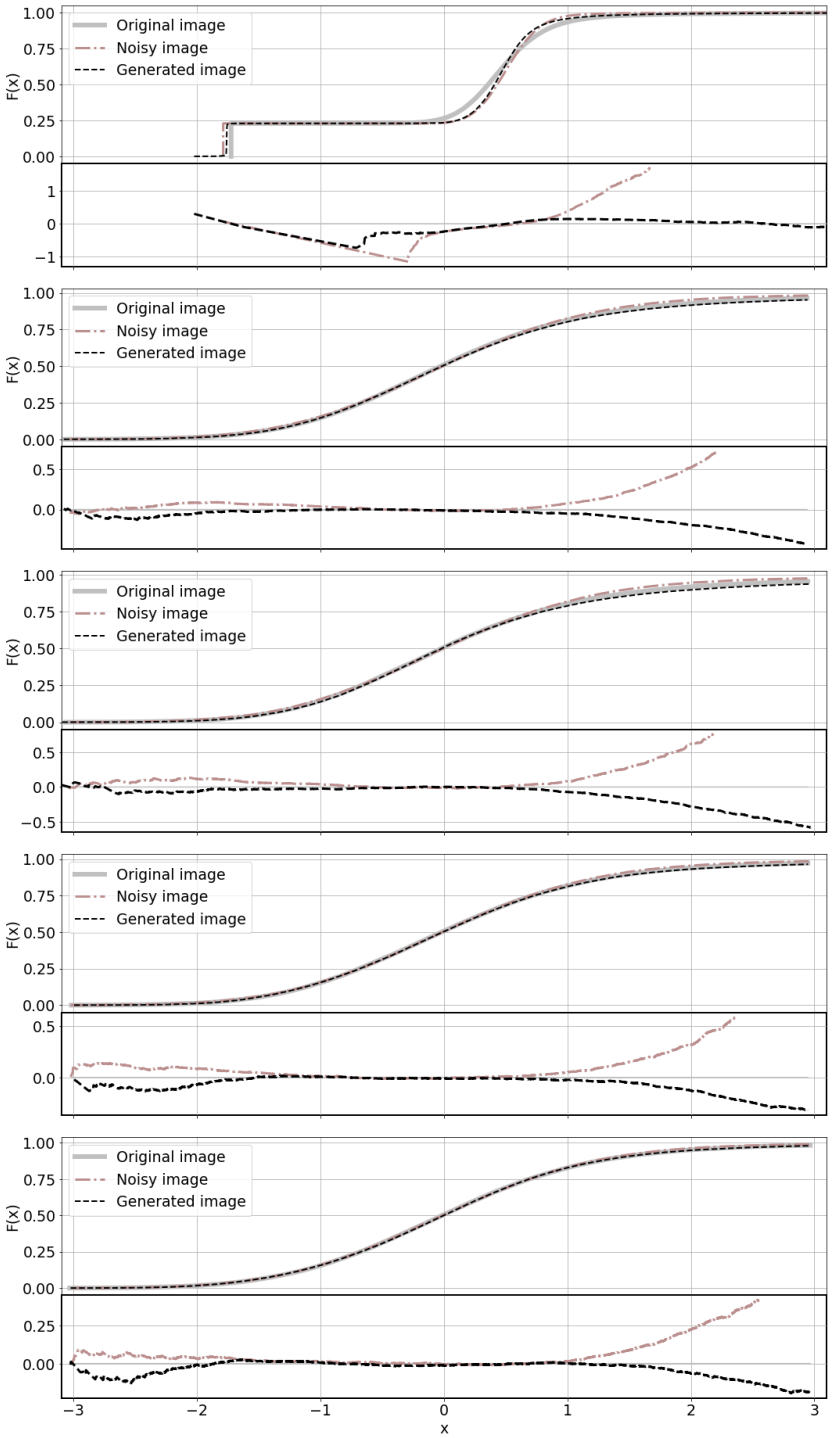

Figure 4.35: The left figure shows normalised CDF and right figure the residual of the of five sub-windows from Figure 4.34

4.2.4 Non-Astronomical image input

To confirm, that the network is not creating artefact as fake stars out of nowhere, we used as input non-astronomical image. The output images are created in two ways. The first is that the image is created as 'Mosaic' (Subsubsection 3.1.2) as all our images. The second one is that the whole image is used as input to the network – denoted 'No Mosaic' (certain creativity problem). Images are scaled by ZScale interval. Note that the network is not trained on this type of data, and in case we generate an image with the same inputs, the results are the same.

The Figure 4.36 shows the outputs of the network with black/white input. In the images, the checker-board pattern is visible. In case of 'No Mosaic' image, the pattern is created by the transpose convolutional layer as explain in Odena et al. [2016]. The pattern on the 'Mosaic' image is a combination of the layer artefact and the way how is the image created. Even though, in both cases the input is an array with just one value the output distributions look different – they are wider when the white image is used as the input. This is caused by the created checker-board pattern, which is the case of 'Mosaic' image sum up and average.

In the Figure 4.37 and Figure 4.38 we can see images with random values with different distributions – Poisson, Normal, Uniform. The result from the image with Normal distribution reminds the distribution of noisy and generated image – original distribution is wider and shifted to higher values. In comparison, the generated image has narrower distribution shifted to the left. In comparison with original distribution, the generated one is more smooth – the standard deviation is smaller than one of the original image.

It is not straightforward to explain the resulting distributions of this experiment. Nevertheless, it proves that not fake stars are created out of random fluctuation on the image.

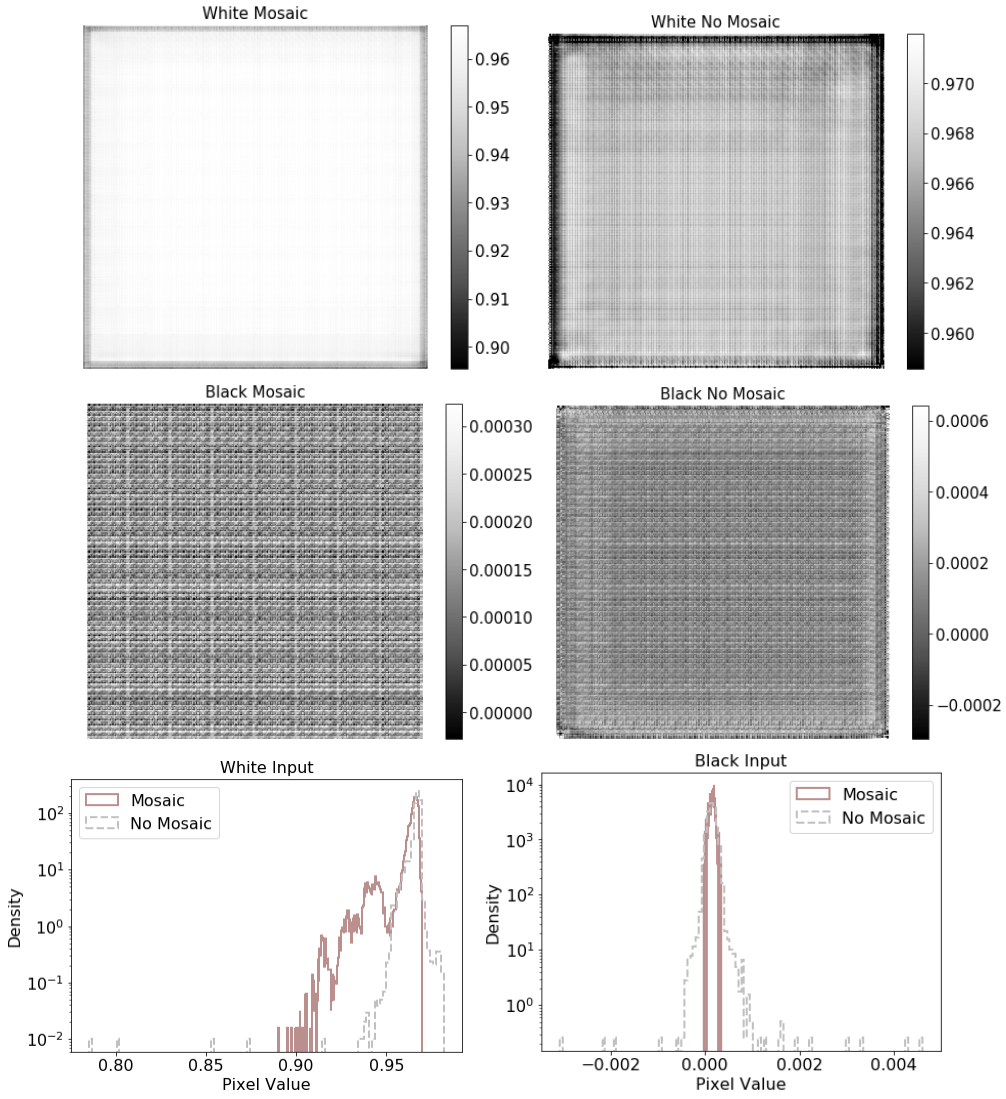

Figure 4.36: The upper row: input is white image or an array full of ones. The left image is created as Mosaic and to create a the right one the whole image is used as an input. The middle row: input is black image or an array full of zeros. The left image is created as Mosaic and to create a the right one the whole image is used as an input. The last row: the pixel distribution of the images.

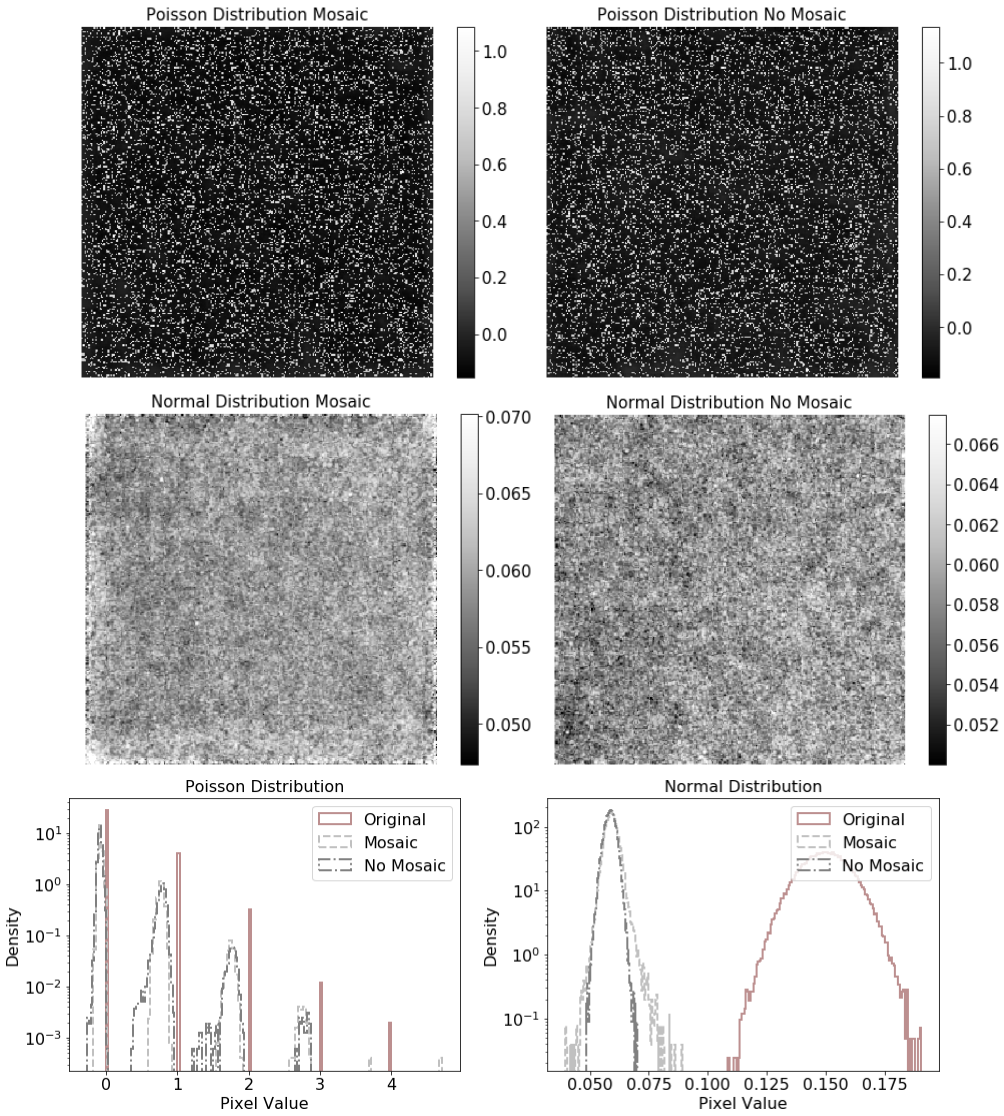

Figure 4.37: The upper row: input is an image with the Poisson distribution . The left image is created as Mosaic and to create a the right one the whole image is used as an input. The middle row: input is an image with normal distribution. The left image is created as Mosaic and to create a the right one the whole image is used as an input. The last row: the pixel distribution of the images.

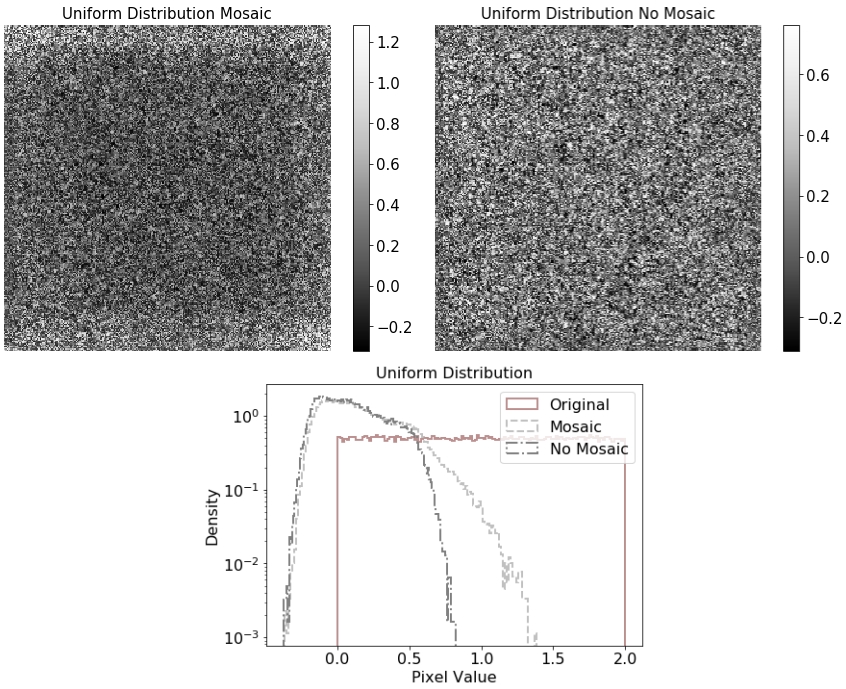

Figure 4.38: Here, as input serves completely image with uniform distributions. The left image is created as Mosaic and to create a the right one the whole image is used as an input.The second row shows pixel distribution of the images.

Figure 4.39: The non-astronomical input of the network. Although there are two stars on the image, the network does not change input image noticeably.

4.2.5 Real data

Even-thought the network is trained on simulated data; we want to apply it on real data to see the network performance. We do not expect good results, and this result is not considered for network evaluation. This section is just an example, where we discuss one image, which means, it is not statistically relevant.
In Table 4.15 we can see results for real data. The input of the network is an image with exposure time 3 sec and is denoted as the short exposure. This image contains `NAN` values which have to be replaced by zeroes because the network is not able to work with, and the network output would contain infinity values afterwards. The long exposure image for comparison is a combination of two images from the same area both with exposure time, 3 sec. Images are combined by Munipack. The number of true positives varies among the used method. The results show that the image generated by the network have better results in star flux and the higher number of detected stars. The Figure 4.41 and Figure 4.42 shows star detection. The Munipack detected more sources than the Star Extractor (also seen in Table 4.15 the True positive column). Because the long exposure image is a bit shifted in comparison with the short exposure and generated image, few sources are missed by the Topcat. The selected maximal distance is 5 pixels. The missed sources can be recognised by blue dashes and green dotted circles close to each other.

The t-test for short exposure time image has results – 1.53 for t-statistic and 0.13 for the p-value. The generated image results are 2.06 and 0.04 respectively. This means that the mean value of long exposure time image is more similar to the short exposure time image. On the other hand, the KS-statistic is better for the generated image 0.09 than for the short exposure time image 0.17. This result is also seen in Figure 4.40 where we can see that CDF of the generated image fits the long exposure time image better.

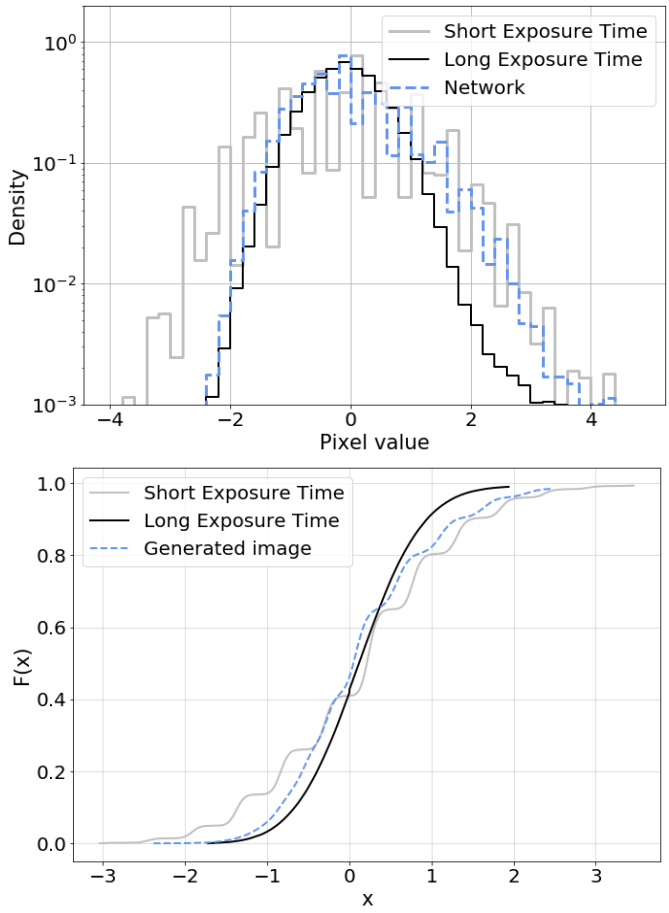

Figure 4.40: The histogram and CDF of the tested images. From the CDF is seen, that down part od the distribution of the generated image is much closer to the long exposure time image than the short exposure one.

Method	Image type	Aperture size [pixel]	Mean error [%]	Median error [%]	True positive	True positive rate [%]
Munipack	G	1	16.4	12.4	55	62.4
Munipack	SE	1	16.7	12.7	45	76.5
Star Extractor	G	10	3.3	2.7	14	82.4
Star Extractor	SE	10	6.1	6.2	13	76.5

Table 4.15: The results of the network for real data. Image type can be G for the generated image or SE for the short exposure time image, which is used as network input. The mean/median error refers to star flux error.

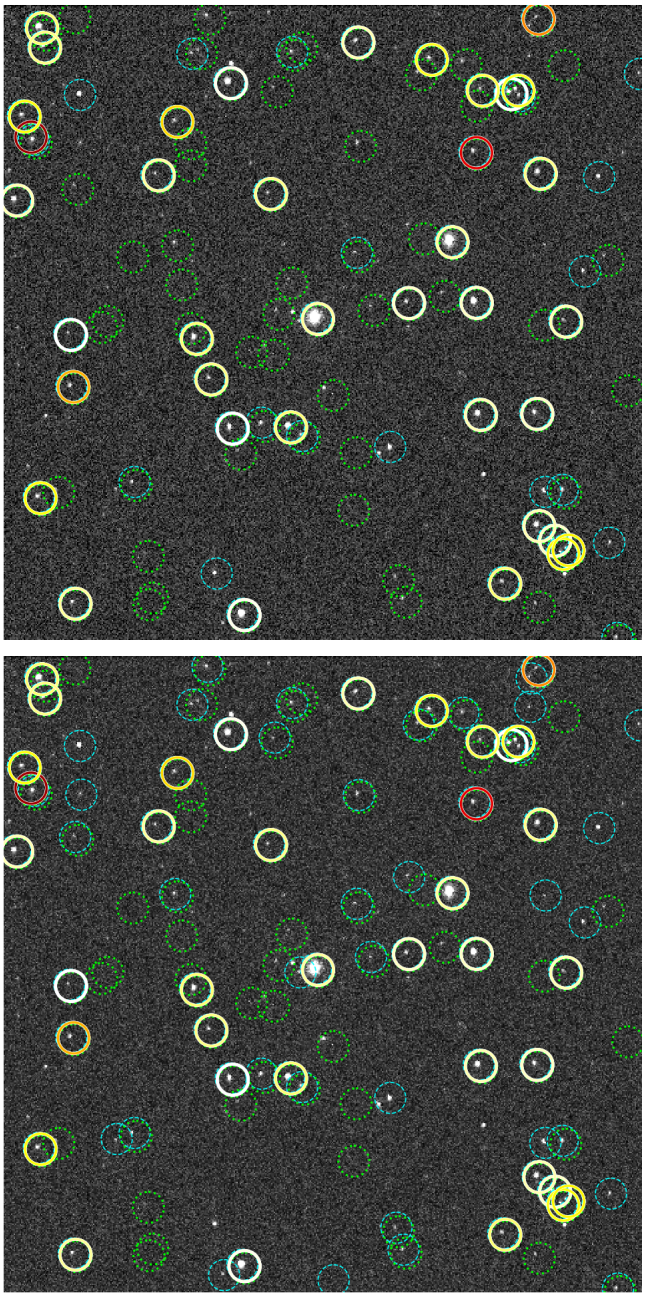

Figure 4.41: Munipack star detection. The upper image is short exposure image and lower image is generated by the network. The green dotted circles are stars detected on the original image but missed on the generated,the blue dashes circles are stars detected just on the generated image. The full circles represent true positive stars and colour of circle indicates flux error.

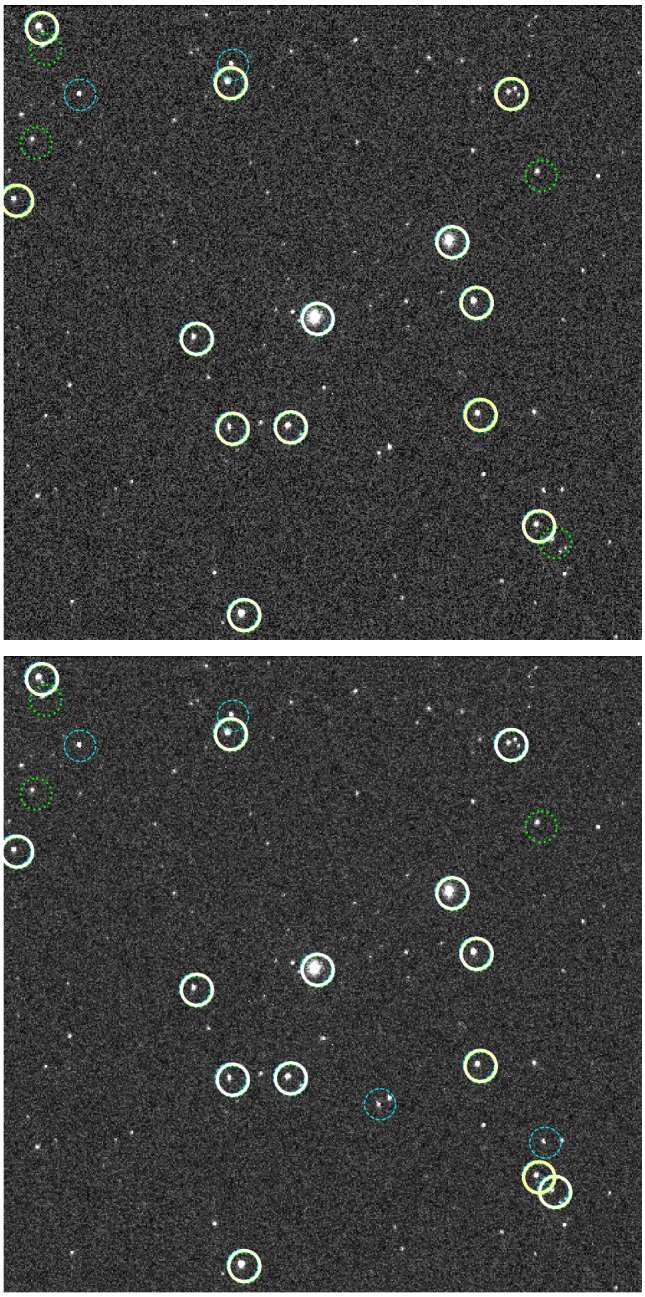

Figure 4.42: Star Extractor detection. The upper image is short exposure image and lower image is generated by the network. The green dotted circles are stars detected on the original image but missed on the generated,the blue dashes circles are stars detected just on the generated image. The full circles represent true positive stars and colour of circle indicates flux error.

4.3 What is inside matters

Because humans are from the nature curious creatures, we want to examine the neural network from the inside. In the beginning the network weights are initiated randomly with uniform distribution. The weights are trainable parameters, which are changed during the training. Examination of weights after training show that distribution of weights in different layers follow the some distribution. In theory the weights should follow the Laplace distribution:

$$\text{Laplace}(\mu, b) = \frac{1}{2b} \exp\left(-\frac{|x-\mu|}{b}\right). \tag{4.2}$$

The distribution is influenced by a loss function, which in this case is L_1. The relationship between the Laplace distribution and L_1 loss function can be shown as: imagine you have input-output pairs $(x_1, y_1) \ldots, (x_N, y_N)$:

$$y_i = \beta_0 + \beta_1 x_{i1} + ... + \beta_p x_{ip} + \varepsilon_i, \tag{4.3}$$

the model is linear regression, where β refers to trainable parameter and ε is noise or uncertainty in the model which has Laplace distribution. This gives rise to a the likelihood function:

$$\prod_{n=1}^{N} \mathcal{L}(y_n|\beta x_n, b). \tag{4.4}$$

Lets take the logarithm of the above expression to find optimal estimates of parameters. Dropping some constants we get:

$$\log \prod_{i=1}^{n} \frac{1}{2b} \exp\left(-\frac{|y_i-\mu|}{b}\right) \tag{4.5}$$

$$\sim \sum_{i=1}^{n} -\frac{|y_i-\mu|}{b}. \tag{4.6}$$

In this expression it becomes apparent why the Laplacian prior can be interpreted as the L_1 loss. The weights in first layer are more important than in latter one, so one should expect less pattern in first layer.

To test this theory, we used the KS-test to compare if weights come from Laplace or the Normal distribution. The distribution of weights in different layers can be seen at Figure 4.44 and Figure 4.45. If we test all filters separately, the KS-test showed that p-value for Network 1 is 0.73 for Laplace and 0.69 for Normal distribution.

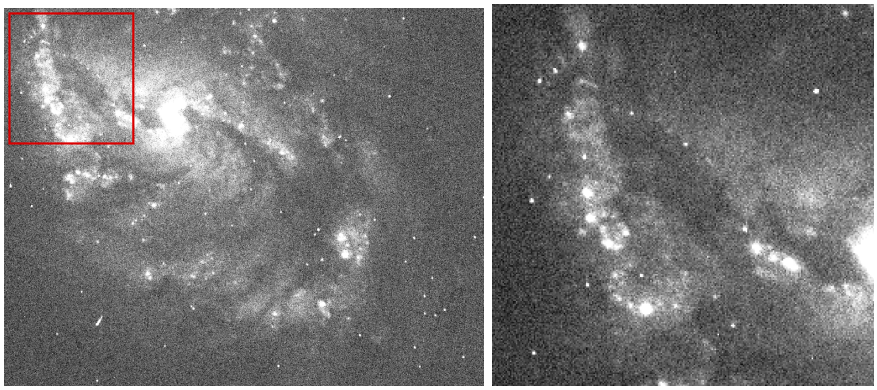

Figure 4.43: Red square is shows part of the image which is the input image for Network 1 in Figure 4.44, 4.46,4.47, 4.48

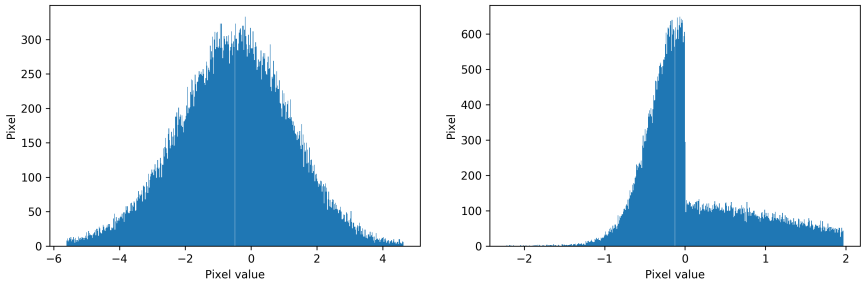

Figure 4.44: Histogram from the first convolutional layer and activation function. Left: histogram of the feature map after convolution. Right: same feature map after LeakyReLU activation function.

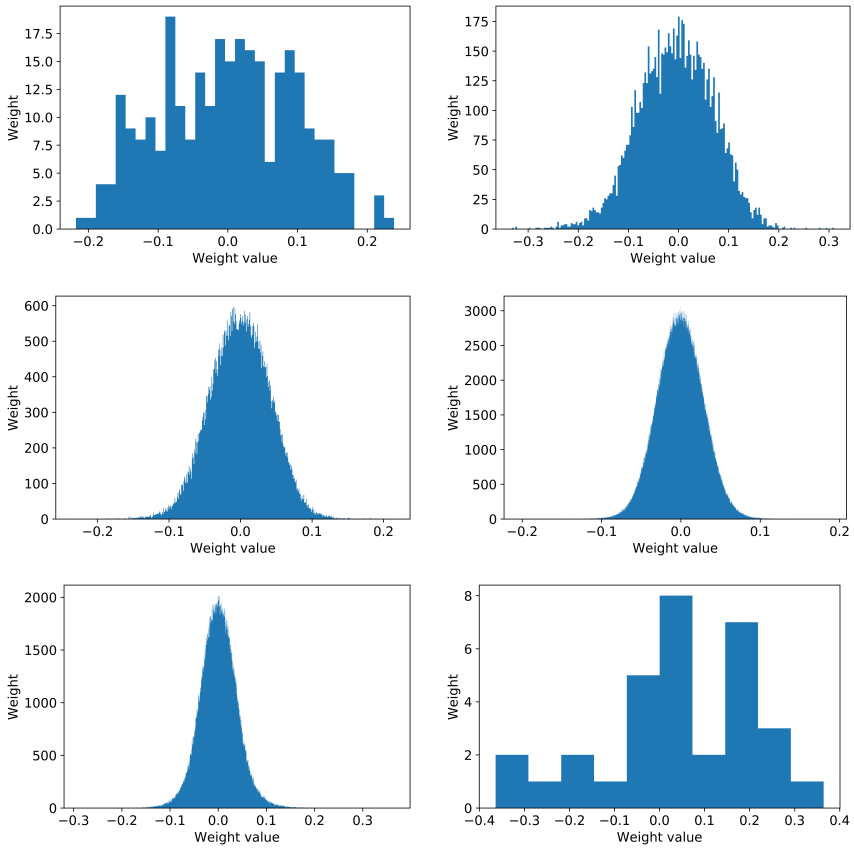

Figure 4.45: Distribution of weights in different layer of Network 1. One histogram corresponds to all weights in one convolutional layer. Upper left figure correspond to the first convolutional layer and right one to second convolutional layer. In the middle are figures for 5. and 9. layer. Last row corresponds to histohrams of 13. and last convolutional layer.

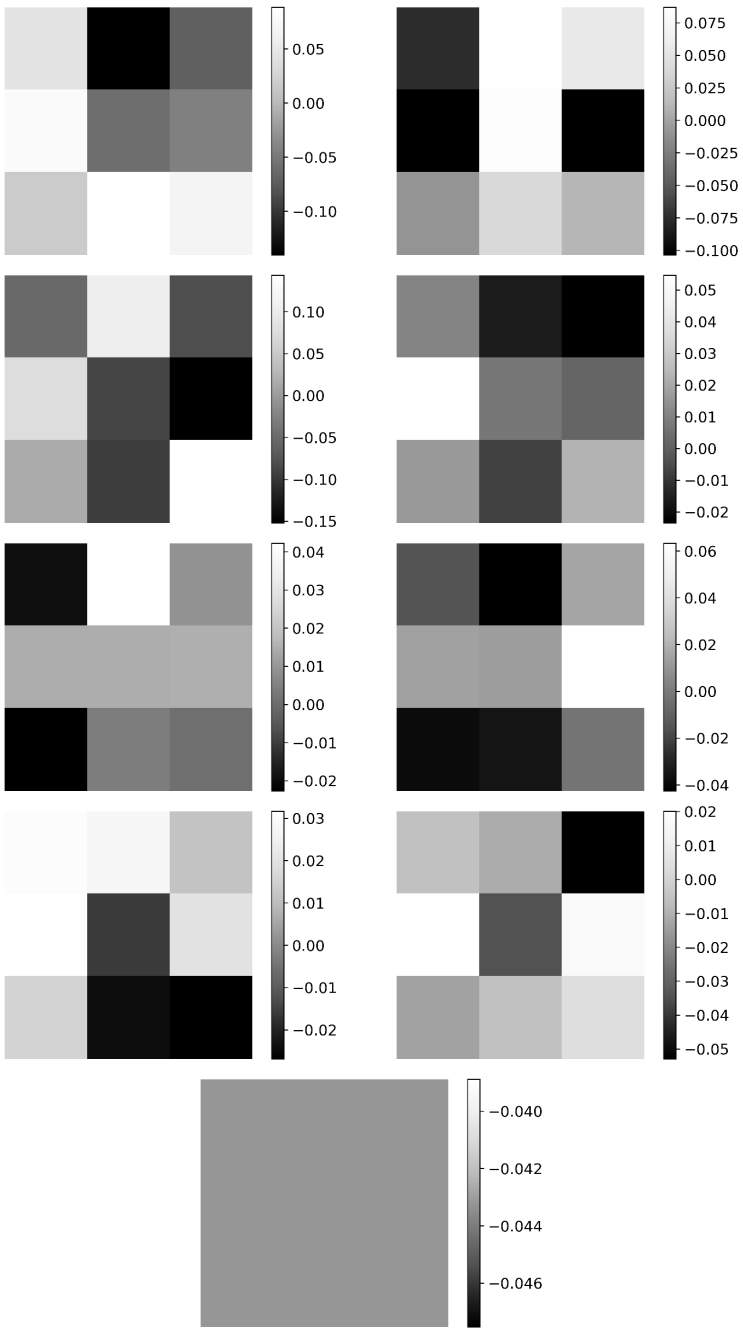

Figure 4.46: Example of weights in Network 1. The down left filter corresponds to the last layer.

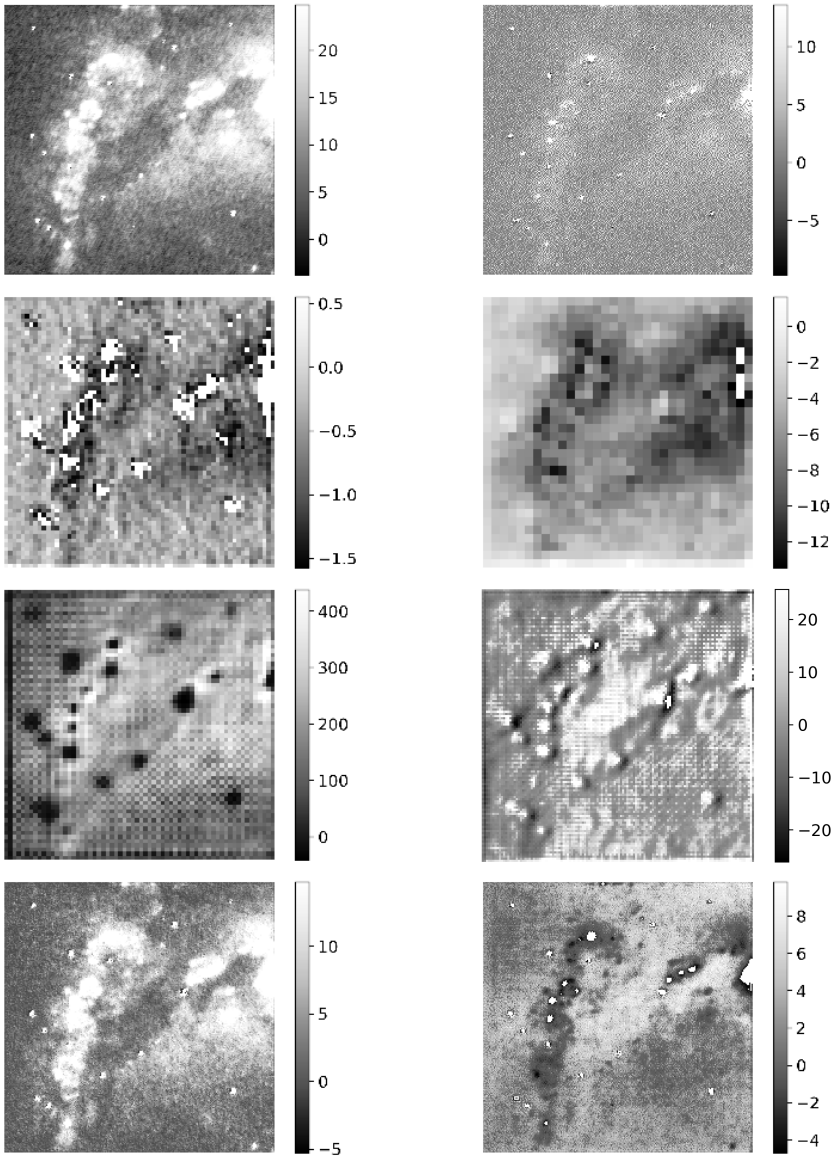

Figure 4.47: Example of convolution feature maps after LeakyReLU activation function from Network 1. Every row corresponds to one convolution layer. From upper left – 1.,2.,5.,7.,13.,16.,18. layer.

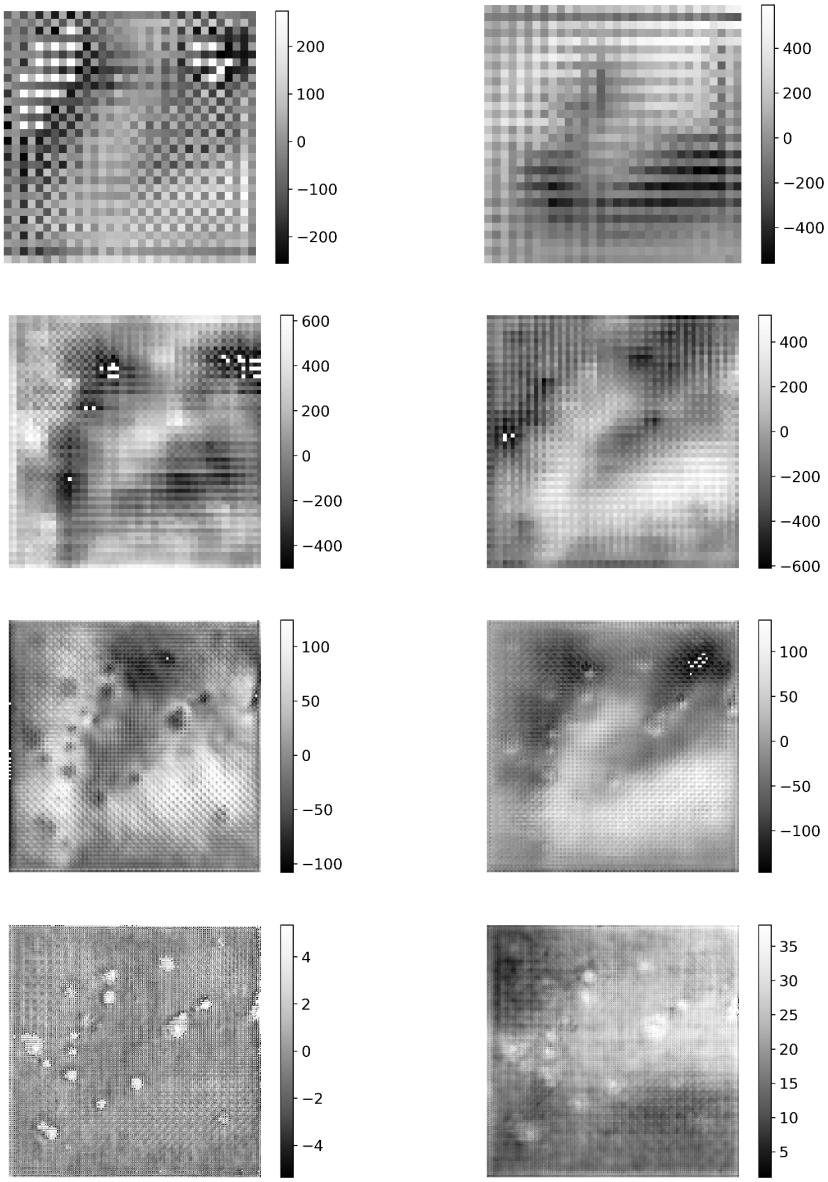

Figure 4.48: Example of transpose convolution feature maps from Network 1. Every row corresponds to one transpose convolution layer

Conclusion and future work

To explore the universe, it is necessary to observe it with different telescopes and create a vast number of astronomical images. To obtain Reducing the noise is an essential and almost mandatory step for further data analysis.

The main idea of our project is to create the convolutional neural network to enhance and denoise astronomical images. The goal of this image processing is to decrease the need for the observational time. For interested readers, the convolutional neural network described in Subsection 2.1. The networks are trained on simulated data with synthetic noise; details about the data can be found in Subsection 3.1. We conduct more than 30 experiments to find suitable architecture (Subsection 4.1). The accuracy of the network is tested by several metrics (Subsection 2.6) – PSNR, SSIM, the star flux error, true positive rate, and different distribution tests. The initial step to finding the best architecture was to examine the image by eye, if there is no artefact and if image by itself makes sense. After that, we combine the loss plot with the PSNR and SSIM to find the best epoch of the training. For the best epoch, the star flux error is made together with a true positive rate and KL-divergence. The rest of the test is done just with the best network describe in Subsection 4.2. The star flux error and the true positive rate is evaluated by two tools – Munipack and Star extractor. The results for the best network are as follows: the mean star flux error is 3.5% with a true positive rate of 95.4%. The test of randomness flux recovery shows that the recovered stars are always within the error assign by employ tool. By examining the image distribution, we find out that the null hypothesis that our generated images come from the same distribution as original ones cannot be disproved. The histograms and the CDF in the Figure 4.21 and Figure 4.22 show agreement between the distribution of the generated and original images. From the normalised CDF and the residuals shown in Figure 4.23 it is obvious that the difference between the original and generated distribution is small. Deeper analysis of the image distribution is available in Subsubsection 4.2.3.

During the evaluation of the network, we also noticed downsides of the network as: *Cosmic rays*. The network can reconstruct cosmic rays as stars. It is good to remove them before using the network. *Edges*. The image edges can contain an artefact learn due to the artefacts at the edges of the training data. *Nan values*. Nan values cause network problems. It is good to change those values before using the network. Like any other platform created for image processing, also this one cannot be used blindly. The network

can be in this point used at real data, although the results are not guaranteed.

In the future, we would like to train the network on real data also in different bands. Our next step is to apply the network on all Hubble space telescope archive collaborating with the ESA Hubble archive. Also, currently (May 2020) another working group is appalling the trained network on the XMM-Newton data. We believe our work helps the scientific community to work more efficiently and to get better results. Use of the machine learning in astrophysics is extensive, and we hope that it brings us to enhanced future in astrophysics!

Bibliography

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

Douglas Adams. The hitchhiker's guide to the galaxy. 1995.

E. Bertin and S. Arnouts. SExtractor: Software for source extraction. , 117: 393–404, June 1996.

Chen Chen, Qifeng Chen, Jia Xu, and Vladlen Koltun. Learning to See in the Dark. *arXiv e-prints*, art. arXiv:1805.01934, May 2018.

Qifeng Chen, Jia Xu, and Vladlen Koltun. Fast Image Processing with Fully-Convolutional Networks. *arXiv e-prints*, art. arXiv:1709.00643, Sep 2017.

François Chollet et al. Keras. `https://keras.io`, 2015.

Pedro Domingos. *The Master Algorithm: How the Quest for the Ultimate Learning Machine Will Remake Our World*. Basic Books, Inc., USA, 2018. ISBN 0465094279.

et al. Dressel, L. "Wide Field Camera 3 Instrument Handbook, Version 12.0" (Baltimore: STScI).

V Dumoulin and F Visin. A guide to convolution arithmetic for deep learning. *ArXiv 1603.07285*, 2016.

et al. Gennaro, M. WFC3 Data Handbook", Version 4.0, (Baltimore: STScI). 2018. URL `https://hst-docs.stsci.edu/display/WFC3DHB/WFC3+Data+Handbook`.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *CoRR*, abs/1502.01852, 2015. URL `http://arxiv.org/abs/1502.01852`.

Filip Hroch. Munipack: General astronomical image processing software, February 2014.

Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual Losses for Real-Time Style Transfer and Super-Resolution. *arXiv e-prints*, art. arXiv:1603.08155, Mar 2016.

Diederik P Kingma and Jimmy Ba. Adam: {A} Method for Stochastic Optimization. *ArXiv 1412.6980*, 2014. URL `http://arxiv.org/abs/1412.6980`.

Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully Convolutional Networks for Semantic Segmentation. *arXiv e-prints*, art. arXiv:1411.4038, Nov 2014.

Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, page 807–814, Madison, WI, USA, 2010. Omnipress. ISBN 9781605589077.

Augustus Odena, Vincent Dumoulin, and Chris Olah. Deconvolution and checkerboard artifacts. *Distill*, 2016. doi: 10.23915/distill.00003. URL `http://distill.pub/2016/deconv-checkerboard`.

William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge University Press, USA, 3 edition, 2007. ISBN 0521880688.

Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Searching for activation functions. *CoRR*, abs/1710.05941, 2017. URL `http://arxiv.org/abs/1710.05941`.

Sebastian Raschka. *Python Machine Learning*. Packt Publishing, 2015. ISBN 1783555130.

Andres C. Rodríguez, Tomasz Kacprzak, Aurelien Lucchi, Adam Amara, Raphaël Sgier, Janis Fluri, Thomas Hofmann, and Alexandre Réfrégier.

Fast cosmic web simulations with generative adversarial networks. *Computational Astrophysics and Cosmology*, 5(1):4, Nov 2018. doi: 10.1186/s40668-018-0026-4.

Kevin Schawinski, Ce Zhang, Hantian Zhang, Lucas Fowler, and Gokula Krishnan Santhanam. Generative adversarial networks recover features in astrophysical images of galaxies beyond the deconvolution limit. , 467(1): L110–L114, May 2017. doi: 10.1093/mnrasl/slx008.

Steven S. Skiena. *The Data Science Design Manual*. Springer Publishing Company, Incorporated, 1st edition, 2017. ISBN 3319554433.

M Taylor. Stilts - a package for command-line processing of tabular data. *ASTRONOMICAL SOCIETY OF THE PACIFIC CONFERENCE SERIES*, 351:666, 06 2006.

Zhou Wang, Alan Bovik, Hamid Rahim Sheikh, and Eero Simoncelli. Image quality assessment: From error visibility to structural similarity. *Image Processing, IEEE Transactions on*, 13:600 – 612, 05 2004. doi: 10.1109/TIP.2003.819861.

Barak Zackay and Eran O. Ofek. How to COAAD Images. I. Optimal Source Detection and Photometry of Point Sources Using Ensembles of Images. , 836(2):187, Feb 2017a. doi: 10.3847/1538-4357/836/2/187.

Barak Zackay and Eran O. Ofek. How to COAAD Images. II. A Coaddition Image that is Optimal for Any Purpose in the Background-dominated Noise Limit. , 836(2):188, Feb 2017b. doi: 10.3847/1538-4357/836/2/188.

Xuaner Cecilia Zhang, Qifeng Chen, Ren Ng, and Vladlen Koltun. Zoom To Learn, Learn To Zoom. *arXiv e-prints*, art. arXiv:1905.05169, May 2019.

H. Zhao, O. Gallo, I. Frosio, and J. Kautz. Loss functions for image restoration with neural networks. *IEEE Transactions on Computational Imaging*, 3 (1):47–57, March 2017. doi: 10.1109/TCI.2016.2644865.

Tiziano Zingales and Ingo P. Waldmann. ExoGAN: Retrieving Exoplanetary Atmospheres Using Deep Convolutional Generative Adversarial Networks. , 156(6):268, Dec 2018. doi: 10.3847/1538-3881/aae77c.