Poisson Denoising of Images
Using Deep Neural Networks
Inspired by Classical Dictionary Based Algorithms

Assaf Mauda
Poisson Denoising of Images
Using Deep Neural Networks
Inspired by Classical Dictionary Based Algorithms

Research Thesis

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science

Assaf Mauda

Submitted to the Senate
of the Technion — Israel Institute of Technology
Heshvan 5782 Haifa October 2021
This research was carried out under the supervision of Prof. Michael Elad, in the Faculty of Computer Science.

**Acknowledgements**

I would like to thank my supervisor, Prof. Michael Elad, for his support and advice throughout this work.

This thesis is dedicated to my wife.

The Technion’s funding of this research is hereby acknowledged.
# Contents

List of Figures

List of Tables

Abstract 1

1 Introduction 3

2 Preliminaries 5
   2.1 Poisson Noise .......................... 5
   2.2 Variance Stabilizing Transformations (VST) ............... 6
   2.3 MAP Estimation .......................... 7
   2.4 MMSE Estimation .......................... 9
   2.5 Iterative Soft Thresholding Algorithm (ISTA) ............... 9
   2.6 Alternating Direction Method of Multipliers (ADMM) .......... 11

3 Previous Work 13
   3.1 Gaussian Denoising Algorithms .......................... 13
      3.1.1 Self-Similarity based methods .......................... 13
      3.1.2 K-SVD Denoising .......................... 14
      3.1.3 Denoising Convolutional Neural Network (DnCNN) .......... 15
      3.1.4 Convolutional Sparse Coding Network for Denoising (CSCNet) 15
   3.2 Classic Algorithms for Poisson Denoising .......................... 17
      3.2.1 Direct methods .......................... 17
      3.2.2 Leveraging Gaussian Denoisers .......................... 18
   3.3 Deep Neural Networks for Poisson Denoising .......................... 21
      3.3.1 DenoiseNet .......................... 21
      3.3.2 VST-NET .......................... 22
      3.3.3 MC²RNet .......................... 23

4 The VST-CSCNet model 25
   4.1 The Model .......................... 25
   4.2 The Dataset .......................... 26
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.3</td>
<td>Model parameters</td>
<td>26</td>
</tr>
<tr>
<td>4.4</td>
<td>Training</td>
<td>26</td>
</tr>
<tr>
<td>4.5</td>
<td>Results</td>
<td>27</td>
</tr>
<tr>
<td>5</td>
<td>The ADMM-CSCNet model</td>
<td>29</td>
</tr>
<tr>
<td>5.1</td>
<td>The Model</td>
<td>29</td>
</tr>
<tr>
<td>5.2</td>
<td>Multiscale ADMM-CSCNet model</td>
<td>30</td>
</tr>
<tr>
<td>5.3</td>
<td>Training</td>
<td>32</td>
</tr>
<tr>
<td>5.4</td>
<td>Results</td>
<td>33</td>
</tr>
<tr>
<td>6</td>
<td>Conclusion</td>
<td>37</td>
</tr>
</tbody>
</table>

Hebrew Abstract

i
# List of Figures

2.1 Gaussian approximations of Poisson distributions .......................... 6  
2.2 Variance stabilization of the Anscombe transform .......................... 7  
2.3 A single LISTA iteration ......................................................... 11  

3.1 DenoiseNet architecture .......................................................... 22  
3.2 VST-NET architecture ............................................................. 23  
3.3 MC²RNet architecture .............................................................. 24  

4.1 The VST-CSCNet model ............................................................ 26  
4.2 Denoising results of the VST-CSCNet model for peak = 2.0 ............... 27  
4.3 Denoising results of the VST-CSCNet model for peak = 0.1 ............... 28  

5.1 The ADMM-CSCNet model iteration ............................................. 30  
5.2 The multiscale ADMM-CSCNet model .......................................... 32  
5.3 Single scale ADMM-CSCNet denoising results for peak = 2.0 ............ 33  
5.4 Single scale ADMM-CSCNet denoising results for peak = 0.1 ............. 34  
5.5 Multi scale ADMM-CSCNet denoising results for peak = 0.1 ............. 35  
5.6 Standard test images .............................................................. 36
List of Tables

5.1 Denoising performance (PSNR, dB) on the BSD68 dataset . . . . . . . . 35
5.2 Denoising performance (PSNR, dB) on standard images . . . . . . . . 36
Abstract

Removing Poisson noise from low-light images is an important and fundamental task in image processing. While most of the research on image denoising focus on Gaussian noise, Poisson denoising is essential for many applications, including night vision, astronomy and fluorescence microscopy. Several approaches exist for this challenging task, some handle the Poisson noise distribution directly, some leverage a Gaussian denoiser and newer deep learning approaches often ignore the specific noise distribution when selecting their model, relying only on a supervised learning strategy.

Leveraging a Gaussian denoiser is mostly done using a variance stabilizing transformation (VST), which transforms a Poisson noisy image into an image with an approximately Gaussian noise. A Gaussian denoiser can then be used, and an inverted transform yields the clean image. While this method is widely used and works well for higher SNR image signals, the Gaussian approximation fails for very low photon-counts, and a different approach should be considered. For example, using the Plug-and-Play Prior method, the Poisson denoising of an image can be performed in an iterative manner using a Gaussian denoiser across the entire image intensity range. Deep learning methods use a large dataset of images to train parametric denoisers and achieve state-of-the-art results. Nevertheless, the interpretability and explainability of these denoisers is lacking. To combat these problems, a possible approach is to adapt classical image processing principles and algorithms into the denoiser architecture.

In this work we propose several novel deep learning networks for Poisson denoising, all relying on classical image processing principles. The proposed solutions are based on sparse representation and multiscale analysis principles, and their action and learned parameters can be explained theoretically. We compare the performance of these models to other common classical and deep learning algorithms and show the competitive results of these models.
Chapter 1

Introduction

The removal of noise from an image is an important and fundamental task in image processing. The statistical distribution of this noise is dependent on the measuring technique and the nature of the captured image and should be considered when tackling the denoising problem at hand. In some applications, such as in night vision, astronomy and fluorescence microscopy, the images are acquired under low light conditions and the image sensor counts a small number of photons for each pixel. Under these circumstances, the noise can be modeled using the Poisson distribution and is called Poisson-noise or shot-noise. The recovery of clean images in these applications is essential, and therefore effective Poisson-denoising algorithms are required.

Several methods can address this problem, the most common one is by approximating the Poisson distribution of the captured image with a Gaussian one, which is reasonable for a high enough photon count in the sensor. In this way, using a variance stabilizing transform (VST) such as the Anscombe transform on the image will result in an image signal with approximately Gaussian noise. Many Gaussian denoising algorithms are available and can then be used to denoise the resulting image. Finally, an inverse Anscombe transform can be applied to yield the resulting denoised image. While this method is widely used and works well for higher SNR image signals, for the very low photon-count case the Gaussian approximation of the noise does not hold, and a different approach should be considered. For example, using the Plug-and-Play Prior method, the Poisson denoising of an image can be performed in an iterative manner using a Gaussian denoiser across the entire image intensity range [RGE16]. Alternatively, a denoising algorithm that directly addresses the Poisson noise can be used.

In recent years, deep learning has achieved state-of-the-art results in many image processing tasks, including image denoising. Nevertheless, the interpretability and explainability of deep learning models is still lacking, and their method of solving complex tasks is unclear. In order to combat these problems, a possible approach is to adapt classical image processing principles and algorithms into the models architectures.

In this work we propose several deep learning models for the purpose of low peak Poisson denoising, based on classical image processing principles. Our models leverage
the CSCNet Gaussian denoiser, which is a deep learning convolutional neural network based on the classical convolutional sparse coding model.

The first model we propose uses the Anscombe VST approach and rely on the CSCNet model as its Gaussian denoiser. This relatively simple model already achieved state-of-the-art results on BSD68 dataset at peak noise of 2.0 but failed on very low peak images.

Our second model uses ADMM in a modified Plug-and-Play prior scheme and achieved excellent results for the very low peak noise value of 0.1, as well as slightly surpassing our VST model at peak noise 2.0.

In addition, we propose another model by incorporating multiscale analysis principles into our ADMM model. This model surpasses the ADMM model, especially for very low peak images, and produces slightly smoother images while preserving edges.

This thesis is organized as follows: In Chapter 2 preliminary knowledge is provided, including the specifics about Poisson noise, Bayesian estimation and optimization methods used throughout this thesis. In Chapter 3 a review of previous work relevant to this thesis is provided, focusing on Gaussian and Poisson denoising algorithms. In Chapter 4 our first model, combining the VST method with the CSCNet denoising architecture, is described and its denoising results are demonstrated. In Chapter 5 our second and third models, using the ADMM approach with the CSCNet architecture in a single-scale or multi-scale fashion are described, and their denoising results are compared to the first model and to previous work in the field. Finally, in Chapter 6 a summary of our models and results is provided, and some open questions are suggested for future research.
Chapter 2

Preliminaries

2.1 Poisson Noise

Natural images always contain some amount of random noise. This noise has a probability distribution that depends on the acquisition method used and the conditions at the time of the image capture, and sometimes on the image itself.

The most common type of encountered noise is the independent and identically distributed (i.i.d) Gaussian noise, sometimes referred to as additive white Gaussian noise (AWGN). Denoting the original image as $x$, the acquired image signal as $y$ and the noise as $n$, the acquired signal can be expressed as:

$$y = x + n,$$

where $n \sim N(0, \sigma^2 I)$, and $\sigma$ is the standard deviation of the noise. This type of noise can result from many processes, among which are thermal noise at the camera sensor.

Another type of noise is the Poisson noise, sometimes referred to as shot noise. This type of noise is present when the sensing process involves counting of discrete events – most notably when measuring currents caused by the movement of a small number of electrons, or when sensing a small number of photons that hit an image sensor. The signal in these cases is the measured number of particles, and its distribution follows the Poisson distribution: Denoting the original image pixel at position $i$ as $x[i]$ and the measured signal at pixel $i$ as $y[i]$, we have:

$$\forall i : y[i] \sim Pois(x[i]),$$

where $Pois(x[i])$ denotes the Poisson distribution with the parameter $\lambda = x[i]$. The likelihood of observing the pixel value $y[i]$ at pixel $i$ given the original image pixel value $x[i]$ is:

$$P(y[i]|x[i]) = \frac{x[i]^{y[i]} \cdot e^{-x[i]}}{y[i]!}.$$
The mean value $\mu_i$ and the variance $\sigma_i^2$ of the Poisson distribution for the signal pixel value $y[i]$ are both equal to $x[i]$, which results in $SNR = \frac{\mu_i}{\sigma_i} = \sqrt{x[i]}$. Therefore, lower intensity images yield noisier signals, and thus it is common to use the peak value of an image to evaluate its noise level.

The Poisson distribution of a pixel value in a high peak image can be approximated as a Gaussian distribution with equal mean and variance. However, for low peak values this approximation does not hold (Figure 2.1) – this situation is important in cases where low intensity images are acquired, such as astronomy [SSC+10], fluorescence microscopy [BKB+10] and PET/SPECT imaging [RSBD08]. Removing the noise in these examples (and many others) is an important image processing task that requires effective and efficient solutions.

![Figure 2.1: Approximation of Poisson distributions with different $\lambda$ values with Gaussian distributions with equal mean and variance.](image)

### 2.2 Variance Stabilizing Transformations (VST)

While the Poisson distribution can be approximated as a Gaussian, its variance is equal to its mean, which implies that in a Poissonian noisy image the variance is not constant across all pixels as in a Gaussian noisy image. However, some transformations, called variance stabilizing transformations (VSTs), can stabilize the variances of each image pixel such that the image noise will become approximately i.i.d. Gaussian. One such VST is the Anscombe transform:

$$A(y) = 2\sqrt{y + 0.375}.$$  \hspace{1cm} (2.4)
As shown in Figure 2.2, this transform stabilizes the variance of high enough pixel values to 1.

![Figure 2.2: The standard deviation of $A(y)$ against the Poisson parameter $\lambda = x$ (the pixel value of the clean image).](image)

We should note, however, that at low pixel values the transformation fails to stabilize the variance.

### 2.3 MAP Estimation

A common approach for the reconstruction of an image signal $x$ based on its noisy measured signal $y$ is using the Bayesian rationale, finding the most probable image $\hat{x}_{MAP}$, given that it generated the signal $y$:

$$\hat{x}_{MAP} = \arg\max_x P(x|y) = \arg\max_x \frac{P(y|x) \cdot P(x)}{P(y)} = \arg\max_x P(y|x) \cdot P(x).$$  \hspace{1cm} (2.5)

It is usually more convenient to minimize the negative logarithm of $P(y|x) \cdot P(x)$ instead of maximizing it directly:

$$\hat{x}_{MAP} = \arg\max_x P(y|x) \cdot P(x) = \arg\min_x -\log P(y|x) - \log P(x).$$  \hspace{1cm} (2.6)

$\hat{x}_{MAP}$ is called the maximum a posteriori estimator (MAP) of the image. While the likelihood $P(y|x)$ of observing the signal $y$ given the clean image $x$ can be determined from the noise distribution and is typically accessible, the prior $p(x) = -\log y$ of the original images has to be taken into account, and this is one of the greatest challenges in image processing.

The chosen prior $p(x)$ is never truly known since we do not have access nor can we process all the possible natural images, so it is always approximated. Many advances in the field of image processing are related to better prior designs that capture the distribution of natural images more accurately. These priors are based on observed
image properties, such as:

- Piece-wise smoothness,
- Self-similarity,
- Scale invariance,
- Sparsity, and
- Low rank.

For an image signal acquired under Poisson noise, the likelihood term $P(y|x)$ is as follows:

$$P(y|x) = P(\{y[i]\},\{x[i]\}) = \prod_i P(y[i]|x[i]) = \prod_i \frac{x[i]^{y[i]} \cdot e^{-x[i]}}{y[i]!}, \quad (2.7)$$

and the negative log likelihood $NLL_{Pois}(x, y)$ is

$$NLL_{Pois}(x, y) = -\log P(y|x) = -\log \prod_i \frac{x[i]^{y[i]} \cdot e^{-x[i]}}{y[i]!}$$

$$= \sum_i x[i] - \sum_i y[i] \log x[i] + \sum_i \log y[i]!.$$ \quad (2.8)

Since we want to minimize $NLL(x, y) + p(x)$ for a specific noisy image $y$, we can ignore the constant term $\sum_i \log y[i]!$, which results in

$$NLL_{Pois}(x, y) = \sum_i x[i] - \sum_i y[i] \log x[i] + \text{const} = 1^T x - y^T \log x + \text{const}, \quad (2.9)$$

where $1$ is a vector with the same size as $x$ and $y$, the log function over the vector $x$ operates elementwise, and the images $x$, $y$ and $1$ are treated as 1D vectors for the calculation of the inner products $1^T x$ and $y^T \log x$. The MAP problem is then:

$$\hat{x}_{MAP} = \arg\min_x 1^T x - y^T \log x + p(x). \quad (2.10)$$

In a similar manner, in the case of Gaussian noise for a signal $y$ with $N$ pixels, the likelihood term is:

$$P(y|x) = \prod_i P(y[i]|x[i]) = \prod_i \frac{1}{\sigma \sqrt{2\pi}} e^{-\frac{(y[i] - x[i])^2}{2\sigma^2}}, \quad (2.11)$$

and the negative log likelihood is

$$NLL_{Gaussian}(x, y) = -\log \prod_i \frac{1}{\sigma \sqrt{2\pi}} e^{-\frac{(y[i] - x[i])^2}{2\sigma^2}}$$

$$= \sum_i \frac{(y[i] - x[i])^2}{2\sigma^2} + N \cdot \log \sigma \sqrt{2\pi}$$

$$= \frac{1}{2\sigma^2} \|y - x\|_2^2 + \text{const}. \quad (2.12)$$
The MAP problem in the case of Gaussian noise is therefore:

\[ \hat{x}_{\text{MAP}} = \arg\min_x \frac{1}{2\sigma^2} \|y - x\|_2^2 + p(x). \] (2.13)

2.4 MMSE Estimation

The MAP estimator is not the only valid way to estimate the clean image \( x \). While it produces the most likely image \( \hat{x}_{\text{MAP}} \) that can result in the noisy measurement \( y \) (under a given prior \( p(x) \) assumption), other approaches try to minimize some distance metric (“error”) between the obtained estimate \( \hat{x} \) and the original clean image \( x \). A commonly used error metric is the \( L_2 \) distance \( \|\hat{x} - x\|_2^2 \). The mean squared error (MSE) of an estimator is defined as the mean \( E_x|y\{\|\hat{x} - x\|_2^2\} \). The minimum MSE estimator (MMSE) is then:

\[
\hat{x}_{\text{MMSE}} = \arg\min_{\hat{x}} E\{\|\hat{x} - x\|_2^2 \mid y\} = \arg\min_{\hat{x}} E\{\|\hat{x}\|_2^2 + \|x\|_2^2 - 2\hat{x}^T x \mid y\} \\
= \arg\min_{\hat{x}} \|\hat{x}\|_2^2 - 2\cdot \hat{x}^T E\{x \mid y\} + E\{\|x\|_2^2 \mid y\}. \tag{2.14}
\]

This is a quadratic expression with respect to \( \hat{x} \) and its minimum is given as

\[ \hat{x}_{\text{MMSE}} = E\{x \mid y\}. \tag{2.15} \]

In other words, given a noisy image \( y \), the MMSE is the mean of all possible clean images \( x \) according to their probability \( P(x \mid y) \).

Since \( P(x \mid y) \propto P(y \mid x) \cdot P(x) \) is complex, its mean is usually intractable, and therefore can only be estimated. Using the known likelihood \( P(y \mid x) \) and our chosen prior \( P(x) \), the MMSE can be estimated with the Monte Carlo method for example, by sampling several options of \( x \) from this distribution and averaging the samples.

2.5 Iterative Soft Thresholding Algorithm (ISTA)

Consider the following optimization problem:

\[ \hat{z} = \arg\min_z \frac{1}{2} \|Dz - y\|_2^2 + \lambda \|z\|_1, \tag{2.16} \]

where \( z \in \mathbb{R}^M \), \( y \in \mathbb{R}^N \), \( D \in \mathbb{R}^{N \times M} \) and \( \lambda \in \mathbb{R}, \lambda > 0 \). This optimization problem is encountered in many areas of signal processing, for example, when using the useful sparsity prior for the natural signals. This prior assumes that the possible signals are produced by a sparse linear combination of atoms from a dictionary \( D \). In other words, each signal \( x \) can be described as \( x = Dz \), where \( z \in \mathbb{R}^M \) is a sparse representation.
vector and the sparsity prior is given using the amount of non-zeros in \( z \): \( p(x) = \lambda \| z \|_0 \).

For a Gaussian noisy signal \( y = x + n \) where \( n \sim \mathcal{N}(0, \sigma^2) \), the MAP problem for the denoising of \( y \) is

\[
\hat{z} = \arg\min_z \frac{1}{2\sigma^2} \| Dz - y \|_2^2 + \lambda \| z \|_0,
\]

This optimization problem is non-convex due to the \( L_0 \) norm, and is therefore often relaxed to use the \( L_1 \) norm, resulting in Equation 2.16.

Using the \( L_1 \) norm, this optimization problem is convex and have a global minimum. The Iterative Soft Thresholding Algorithm (ISTA) is an iterative method for finding the optimal \( \hat{z} \) using the following steps until convergence:

\[
z_0 = D^T y \\
z_{n+1} = S_\tau(z_n - \frac{1}{c} D^T (Dz_n - y)),
\]

where \( S_\alpha \) is the soft thresholding operator applied elementwise on its input vector, such that for each entry \( i \):

\[
S_\alpha(z[i]) = \begin{cases} 
  z[i] - \alpha & z[i] > \alpha \\
  0 & |z[i]| \leq \alpha \\
  z[i] + \alpha & z[i] < -\alpha 
\end{cases}
\]

and \( c \in \mathbb{R} \) is a constant satisfying \( c \geq \lambda_{\text{max}} \{ D^T D \} \).

ISTA can be derived in several ways, among which is the majorization-minimization method, or the proximal gradient descent algorithm, and it is guaranteed to converge ([DDDM04], [CW05], [FN03]). Nevertheless, ISTA has a relatively slow convergence rate of \( O(1/t) \). In order to overcome this limitation several algorithms were developed, among them is the notable FISTA algorithm [BT09], which incorporates Nesterov’s accelerated gradient descent into the ISTA framework, and improves the convergence rate to \( O(1/t^2) \) while keeping the method relatively simple.

ISTA can also be performed as a part of a neural network, with LISTA – Learned Iterative Soft Thresholding [GL10]. Equation 2.18 can be constructed as a model inside a neural network using linear layers with shared weights that multiply their input by \( D \) or \( \frac{1}{c} D^T \), biases of \( -y \) and \( x_n \), and the non-linear soft thresholding activation function \( S \). The ISTA algorithm can be unrolled in a neural network using several of these models with shared parameters \( (D, c \text{ and } \lambda) \), and therefore making the network solve Equation 2.16 – while learning \( D, c \) and \( \lambda \) during the training. For faster convergence, \( D \) and \( D^T \) can be separated into two different dictionaried, \( A \) and \( B \), resulting in the following iteration:

\[
z_{n+1} = S_\tau(z_n - A^T (Bz_n - y)).
\]
A single iteration of the LISTA process is depicted in Figure 2.3.

Figure 2.3: A single LISTA iteration.

2.6 Alternating Direction Method of Multipliers (ADMM)

Consider the following common optimization problem for $x \in \mathbb{R}^N$:

$$\hat{x} = \arg\min_x f(x) + g(x). \quad (2.21)$$

The general MAP optimization problem fits this type of problem, where $f$ and $g$ are the likelihood and the prior terms respectively. In many cases it is not easy to solve this problem directly because of the complexity of dealing with $f$ and $g$ together.

The Alternative Direction Method of Multipliers (ADMM) offers an iterative method for the solution of this optimization problem where in each step only one function among $f$ and $g$ is considered, thus producing simpler step.

The solution to the above unconstrained optimization problem is the same as the solution of the following constrained optimization problem:

$$\hat{x}, \hat{v} = \arg\min_{x,v} f(x) + g(v) \quad \text{s.t.} \quad x = v, \quad (2.22)$$

such that $\hat{x} = \hat{v}$. ADMM propose a method to solve this constrained optimization problems of this type using the augmented Lagrangian technique.

The Lagrangian of the problem is:

$$L(x, v, \beta) = f(x) + g(x) + \beta^T(x - v), \quad (2.23)$$

where $\beta$ is the Lagrange multipliers vector. Using the following penalty function:

$$\varphi_{p,\beta}(t) = \frac{p}{2}t^2 + \beta t, \quad \varphi_{p,\beta}'(t) = pt + \beta, \quad (2.24)$$
the augmented Lagrangian is:

\[
F_p(x, v, \beta) = f(x) + g(v) + \varphi_{p, \beta}(x - v) \\
= f(x) + g(v) + \beta^T(x - v) + \frac{p}{2} \|x - v\|_2^2 \\
= f(x) + g(v) + \frac{p}{2} \left\| x - v + \frac{1}{p} \beta \right\|_2^2 - \frac{1}{2p} \|\beta\|_2^2. 
\] (2.25)

The augmented Lagrangian method solves Equation 2.22 using the following iteration scheme:

\[
x_{k+1}, v_{k+1} = \arg\min_{x, v} F_p(x, v, \beta_k) \\
\beta_{k+1} = \varphi'_{p, \beta_k}(x_{k+1} - v_{k+1}) = \beta_k + p(x_{k+1} - v_{k+1}). 
\] (2.26)

This method is composed of an unconstrained optimization problem and a simple update to the Lagrange multipliers \(\beta_k\) in each iteration. In the ADMM scheme, the first optimization problem is replaced by two separate optimization problems – one for \(x\) and one for \(v\), separately. This usually does not yield the optimal \(x_{k+1}, v_{k+1}\), but it is enough for conversion under some reasonable conditions (\(f\) and \(g\) should be closed, proper and convex, [BPC+11]). The ADMM scheme is therefore:

\[
x_{k+1} = \arg\min_{x} F_p(x, v_k, \beta_k) = \arg\min_{x} f(x) + \frac{p}{2} \left\| x - v_k + \frac{1}{p} \beta_k \right\|_2^2 \quad (2.28) \\
v_{k+1} = \arg\min_{v} F_p(x_{k+1}, v, \beta_k) = \arg\min_{v} g(v) + \frac{p}{2} \left\| v - x_{k+1} - \frac{1}{p} \beta_k \right\|_2^2 \quad (2.29) \\
\beta_{k+1} = \beta_k + p(x_{k+1} - v_{k+1}). \quad (2.30)
\]

Denoting \(\mu_k = \frac{1}{p} \beta_k\), this scheme is simplified to:

\[
x_{k+1} = \arg\min_{x} f(x) + \frac{p}{2} \left\| x - v_k + \mu_k \right\|_2^2 \quad (2.31) \\
v_{k+1} = \arg\min_{v} g(v) + \frac{p}{2} \left\| v - x_{k+1} - \mu_k \right\|_2^2 \quad (2.32) \\
\mu_{k+1} = \mu_k + x_{k+1} - v_{k+1}. \quad (2.33)
\]

In this way, \(f\) and \(g\) can be minimized separately, which usually reduces the complexity of the problem, at the cost of having to perform multiple iterations until convergence. This is especially beneficial when the optimization problem for \(x_{k+1}\) or for \(v_{k+1}\) is simple and has a closed form solution. For example, if \(f\) or \(g\) are separable, the optimization problem for \(x_{k+1}\) or for \(v_{k+1}\) is also separable and can usually be solved efficiently. The convergence rate of ADMM is quite slow, but adequate results are usually achieved after only a few iterations for most practical uses without much improvement for further iterations, which makes this method suitable for practical use.
Chapter 3

Previous Work

Cleaning a Poisson noisy image is the subject of many research papers due to its practical uses. The denoising methods can usually be categorized in several ways. Some methods attempt to denoise a Poisson-distributed noisy image directly while considering its specific noise distribution, while other methods attempt to leverage Gaussian denoisers, which are more common and extensively studied, for the task of Poisson denoising by using some sort of transformation. In addition, Poisson denoising can also be tackled using convolutional neural networks and supervised learning, which can show remarkable results even without a direct consideration of the Poisson noise distribution. In this chapter we present a review of Gaussian and Poisson denoising methods that are of direct relevance to our research.

3.1 Gaussian Denoising Algorithms

3.1.1 Self-Similarity based methods

Natural images often possess self-similarity characteristics, i.e., different areas in the image resemble each other. This property is widely used in denoising algorithms, for example, in the NLM [BCM05] and the BM3D [DFKE07] algorithms discussed below.

Non-Local Means (NLM)

The NLM algorithm cleans each pixel $i$ by looking at a small patch $v_i$ around this pixel and finding “similar” patches $\{u_{i,j}\}_{j=1}^{K}$ in the noisy image, where patches are considered similar based on their $L_2$ distance to $v_i$. A weight is given for each patch based on its $L_2$ distance to $v_i$ such that low $L_2$ distance, which indicates a high similarity to $v_i$, results in high weight. A weighted average of the patches $v_i$ and $\{u_{i,j}\}_{j=1}^{K}$ is then calculated, and the cleaned pixel $i$ is extracted from it. This procedure is repeated for all the pixels in the image to afford the cleaned image.

This method relies on the fact that the negative log likelihood of two similar patches in a Gaussian noisy image is proportional to the $L_2$ distance between the patches.
the case of Poisson noisy image, the similarity and the weights between different patches should not be based on the $L_2$ distance, but rather on a negative log likelihood term resulting from the Poisson distribution, as can be seen in Section 3.2.1.

**Block-Matching and 3D filtering (BM3D)**

The BM3D algorithm builds upon the NLM approach of using similar noisy patches for the denoising but adds a sparsity prior instead of just using a weighted average. For each noisy patch, several similar patches are found within the noisy image (in the same way as in NLM, but here it is referred to as block-matching), and the noisy patches are stacked together to a 3D tensor. This tensor is then cleaned by transforming it by the 3D-DCT, applying hard thresholding on the coefficients, and transforming back using the inverse 3D-DCT transform – this process is denoted as collaborative filtering. The cleaned patch is then extracted from this tensor, the procedure is repeated for all patches, and the resulting cleaned patches are averaged to yield an initial estimate for the original image. The same process is then repeated using a Wiener filter instead of a hard thresholding, using the initial estimate for the block-matching. This second stage relies on better block-matching, since it uses a better estimate for the original image, and therefore produces a better reconstruction result.

**3.1.2 K-SVD Denoising**

The sparse and redundant representations model is a well-studied model with wide use in many signal processing fields. It attempts to describe a set of natural signals $W \subset \mathbb{R}^N$ using a small set of atoms $\{d_i\}_{i=1}^K$ (where $d_i \in \mathbb{R}^N$), referred to as a dictionary. This model assumes that each signal in $W$ is formed from a linear combination of a small number of atoms from the dictionary. The dictionary can be represented as a matrix $D \in \mathbb{R}^{N \times K}$ where each column is an atom from the dictionary, and from the model assumption, for each signal $x \in W$ there exist a sparse vector $\alpha \in \mathbb{R}^K$ such that $x = Da$. The redundancy part of the model is in the dictionary size $K$ which is usually larger than $N$ to allow for a “rich” set of natural signals $W$. Given a noisy signal $y = x + n$, several sparse coding algorithms exist for the purpose of recovering the original sparse representation $\alpha$, such as Orthogonal Matching Pursuit (OMP) and Basis Pursuit (BP), with theoretical guarantees for the successful recovery ([DE02], [GN03], [Tro04], [DET06], [Don05], [BHEE10] and more). Recovering the cleaned signal is then achieved by multiplying with the dictionary $D$.

Natural images are usually high dimensional (about $10^6$), and their dimension is even not constant, which makes the direct use of the sparse and redundant model impractical. Therefore, patch-based models such as the K-SVD denoising can be used [EA06]. By dividing the image into $P$ overlapping patches of a small size, the sparse and redundant representations model can be used to clean the patches, which are all from a smaller, constant dimension. Then, the cleaned image can be constructed from the
cleaned patches by overlaying and averaging. The K-SVD algorithm offers an algorithm for learning a dictionary $D$ that will result in the sparsest representations $\{\alpha_i\}_{i=1}^{P}$ of the patches, and uses this dictionary and representations for the construction of the cleaned image. Two different approaches were presented, offering to learn the dictionary based on either the patches of the noisy image to be cleaned, or on clean patches of a dataset of different clean images. Using the patches of the noisy image to be cleaned gives better results, at the cost of increased inference time.

3.1.3 Denoising Convolutional Neural Network (DnCNN)

The advances in the field of deep neural networks, as well as the availability of large image datasets and powerful GPUs, enabled the application of these models in various image processing tasks, including image denoising. In [ZZC+16], a deep convolutional neural network architecture was devised for the Gaussian denoising of images, and for other tasks as well, such as blind Gaussian denoising (denoising an image with a Gaussian noise of an unknown standard deviation), super resolution and JPEG image deblocking. The network architecture is composed of 17 convolutional layers, with ReLU activation function after the first layer and batch normalization and ReLU activation function after each of the next 15 layers. Each layer consists of 64 filters, except for the last layer which has one layer for grayscale images and three layers for RGB images. Overall, the network has $5.55 \times 10^5$ trainable parameters, and is trained using stochastic gradient descent (SGD) with momentum. The model achieved state-of-the-art results for the time for Gaussian denoising, at a significant speed-up due to the use of a GPU – enabling blind Gaussian denoising of a $512 \times 512$ image at 60 ms.

3.1.4 Convolutional Sparse Coding Network for Denoising (CSCNet)

As was previously discussed for K-SVD denoising, the sparse and redundant representations model is not compatible directly with natural images due to their high and non-constant dimension. While the K-SVD model solves this by dividing the image into small patches of constant size, using a local patch cleaning and global averaging, the Convolutional Sparse Coding (CSC) approach is to use a convolutional dictionary for the entire image with several small filters ([GRKN07], [SKL10]).

Under the CSC model, an image $x \in \mathbb{R}^N$ (shown here as 1D for simplicity, the description holds also for 2D) is described using:

$$x = \sum_{i=1}^{m} d_i \ast Z_i,$$  \hspace{1cm} (3.1)

where $N$ is the overall image dimension (in the order of $10^6$), $m$ is the number of filters, $\{d_i\}_{i=1}^{m}$ are the filters and $\{Z_i\}_{i=1}^{m}$ are sparse feature maps. A global dictionary $D$ can be constructed from the concatenation of the $m$ convolutional circulant matrices of
the filters \(\{d_i\}_{i=1}^{m}\), and a global representation \(\Gamma\) can be constructed from the \(m\) sparse feature maps \(\{Z_i\}_{i=1}^{m}\), such that:

\[
x = D\Gamma.
\] (3.2)

The dictionary \(D\) is very large, with a size of \(N \times mN\), but multiplying by \(D\) is equivalent to applying \(m\) different convolutions with small filters, and thus practical. In addition, the filters \(\{d_i\}_{i=1}^{m}\) are enough to describe \(D\) for any image size, making the convolutional dictionary approach flexible for different image sizes. In contrast to most patch-based approaches, the CSC model combines the local processing (using small filters) with the global treatment (by looking at sparsity over the entire image at once).

Using the CSC model, the MAP problem for Gaussian image denoising of the signal \(y\) is:

\[
\hat{\Gamma}_{MAP} = \arg\min_{\Gamma} \frac{1}{2} \|D\Gamma - y\|_2^2 + \lambda \|\Gamma\|_p,
\] (3.3)

where \(p \in \{0, 1\}\). In addition, the parameter \(\lambda > 0\) can have different values for different filters. Several methods exist for solving this problem with some theoretical guarantees [PSE17].

While the classic CSC model has shown success in some image processing tasks ([ZSE19], [PRES17], [GZX+15], [LCWJW16], [ROSE18]), it was used mostly for image textures. In contrast, for the image denoising problem the model failed to produce good results.

The CSC model was used for Gaussian image denoising in [SG17] by implementing LISTA [GL10] in a convolutional neural network to solve the MAP problem above with separate dictionaries \(A\) and \(B\). A different convolutional dictionary \(C\) was then used to reconstruct the cleaned image, and the network was trained in a supervised manner. The results were on-par with classic K-SVD denoising, but not as good as deep K-SVD [SEM19], where the patch-based K-SVD denoising algorithm was unrolled into a neural network, or as DnCNN [ZZC+16].

Two main disadvantages of the CSC model were identified in [SE19]: First, the CSC model has high mutual coherence \(\mu(D)\). Since natural images usually contain large areas of piecewise smooth regions, the dictionary should contain piecewise smooth filters. The autocorrelation of these filters under small spatial movements is high, which results in high mutual coherence for the dictionary. A theoretical guarantee for the recovery of the true representation \(\Gamma\) was obtained in [PSE17]:

\[
\|\Gamma\|_{0,\infty} < \frac{1}{2} \left( 1 + \frac{1}{\mu(D)} \right).
\] (3.4)

For high mutual coherence, the theoretical guarantee does not hold. Second, the CSC model employs a MAP estimation approach, and not an MMSE approach like patch-
based K-SVD. Therefore, when measuring performance in terms of MSE, it is not optimal.

These problems were addressed in [SE19] by using a strided convolutional dictionary. Using strides of more than one pixel, the autocorrelation between spatially adjacent atoms is significantly decreased, and the mutual coherence of the dictionary can be lowered. This gives a stronger theoretical guarantee for the successful recovery of the sparse representation.

In addition, there are several options for the initial offset of the strided convolutions. Each initial offset represents a different support $S$ of the non-strided dictionary and yields a different cleaned image $\hat{x}_{MAP,S}$. The MMSE estimator for the clean image can be written as:

$$\hat{x}_{MMSE} = E\{x|y\} = E_S \{E\{x|y,S\}\} = E_S\{\hat{x}_S\},$$  \hspace{1cm} (3.5)

where $\hat{x}_S$ is the MMSE estimator of the clean image under the specific support $S$. We can approximate $\hat{x}_S$ with $\hat{x}_{MAP,S}$ and approximate the mean over all the supports by averaging over all the possible initial offsets $\Theta$:

$$\hat{x}_{MMSE} \approx \sum_{S \in \Theta} P(S) \cdot \hat{x}_S \approx \frac{\sum_{S \in \Theta} \hat{x}_{MAP,S}}{|\Theta|}. \hspace{1cm} (3.6)$$

In other words, averaging the cleaned outputs from all the different initial offsets results in an approximated MMSE estimator, which should result in better MSE than for a MAP estimator.

The resulting model, named CSCNet, was also unrolled into a neural network and trained in a supervised manner. Using only 63.7K parameters, CSCNet model gave the same performance in terms of PSNR as DnCNN, which uses 555K parameters.

### 3.2 Classic Algorithms for Poisson Denoising

#### 3.2.1 Direct methods

**Poisson Non-Local Means (Poisson NLM)**

The non-local means (NLM) method uses the self-similarity prior for natural images to clean Gaussian noisy images by averaging similar patches. This property can be especially useful for very low SNR image denoising, where the noisy image does not contain a lot of information. In the NLM denoising for Gaussian noisy images the similarity between the patches is determined based on the $L_2$ norm, which can be shown to be proportional to the probability that two Gaussian noisy patches resulted from identical clean patches. For other noise distributions such as the Poisson distribution, this similarity metric is not optimal. The usual solution is to get an estimate for the clean image by using a rough denoiser and determine patch similarities based on the
patches of this estimate. A better approach, used in [DTD10] in the Poisson NLM algorithm, is to adjust the distance metric to the Poisson distribution:

$$\text{dist}_{\text{Poisson}}(y_1, y_2) = y_1 \log y_1 + y_2 \log y_2 - (y_1 + y_2) \log (y_1 + y_2),$$  \hspace{1cm} (3.7)

where $y_1$ and $y_2$ are two pixel values in the noisy image, and $\text{dist}_{\text{Poisson}}(y_1, y_2)$ is the negative log-likelihood for the pixels resulting from the same underlying clean image pixel values. The results showed a PSNR improvement of 1.0 to 2.0 dB over the original NLM algorithm, and 0.1 to 0.8 dB over NLM using an estimate for the clean image (derived using the original NLM algorithm).

**NLSPCA and SPDA**

A complicating factor of Poisson distributed images is the non-negativity constraint on the image pixel values. This constraint encouraged the authors of the Poisson Non-Local Sparse PCA (NLSPCA) method [SHDW14] and of the Sparsity Poisson Denoising Algorithm (SPDA) [GE14] to use an exponential dictionary model for the image patches, which eliminates the constraint:

$$y_i = \exp(D\alpha_i),$$  \hspace{1cm} (3.8)

where $y_i$ is a noisy image patch, $D$ is a patch dictionary, $\alpha_i$ is a sparse representation vector and the exponent function operates elementwise. In addition to a sparsity prior for the image patches, both methods use a self-similarity prior by clustering similar image patches together and operating on each cluster separately. The NLSPCA method uses K-means to cluster the image patches to a small number of large clusters, learns a dictionary for each cluster and uses it to clean the patches in the cluster. In contrast, the SPDA algorithm learns a single dictionary for all the patches, and clusters the patches using a greedy algorithm that aims to produce many clusters with a small and similar size. In addition, it incorporates self-similarity using joint sparsity, by forcing all the patches in each cluster to have the same support. Both methods gave better PSNR values than the Poisson NLM methods, with SPDA surpassing NLSPCA by about 0.4 to 0.9 dB for low peak values and more than 1.2 dB for higher peak values.

### 3.2.2 Leveraging Gaussian Denoisers

**Variance Stabilizing Transformation (VST) based methods**

Since Gaussian denoisers are very well studied, it is tempting to use them also for the task of Poisson denoising. Afterall, the Poisson distribution can be approximated as a Gaussian for high enough peak values (Figure 2.1). Unfortunately, this is not enough, since the variance of a Poisson noisy image is dependent on the image itself, while for Gaussian noisy image it is constant.
Using a Variance Stabilizing Transform (VST) such as the Anscombe transform $A(y) = 2\sqrt{y + 0.375}$ on the pixel values of the noisy image $y$, their variance can be stabilized to around 1, and the resulting image will have approximately Gaussian i.i.d noise. A Gaussian denoiser can then be used to clean the image, followed by the inversed Anscombe transform.

The algebraic inverse $A^{-1}(y) = (0.5y)^2 - 0.375$ introduces a bias to the cleaned image and is therefore inadequate. The exact unbiased inverse should be used instead [MF11b], a closed-form approximation of which is given in [MF11a]:

$$A_{\text{unbiased,approx}}^{-1}(y) = \frac{1}{4}y^2 - \frac{1}{8} + \frac{1}{4}\sqrt{\frac{3}{2}y} - \frac{11}{8}y^{-2} + \frac{5}{8}\sqrt{\frac{3}{2}y^{-3}}. \quad (3.9)$$

This method for Poisson denoising is widely used for higher peak noise images, where the transformation can stabilize the variance successfully, and gives good results. However, for lower peak noise values, the transformation fails to stabilize most of the pixel values, and the denoising results are poor. Some methods can help dealing with these cases, such as binning and iterative VST.

Binning is a widely used practice for handling very low SNR Poisson noisy images. It involves decreasing the image resolution by dividing it into pixel bins of size $n \times n$, and treating the bins as pixels of a new, low resolution image (downsampled by a factor of $n$), containing the total number of photons that arrived into these bins. In this way, the low-resolution image has a Poisson noise distribution with a peak value $n^2$ times higher than the peak value of the original image, and the VST method can then be used to clean it successfully. The last stage is upscaling or super resolution of the obtained image to the original image size. In this way, spatial accuracy is exchanged for higher SNR in order to better clean a very low SNR image.

The iterative VST scheme [AF16] starts by performing the original VST method with BM3D as its Gaussian denoiser (denoted as I+VST+BM3D) to get an estimate for the clean image. Using the noisy image and the obtained clean image estimate, a more accurate VST is then used for better variance stabilization, especially for lower pixel values. The transformed image is then cleaned with BM3D and transformed back to obtain the cleaned image. This method produces far better PSNR values for low peak Poisson denoising than the original, single iteration of VST+BM3D, with 1.0 to 3.0 dB improvement, depending on the specific peak value.

The Plug-and-Play Prior method

The MAP optimization problem for Poisson image denoising is

$$\hat{x}_{MAP} = \arg\min_x NLL_{\text{Pois}}(x, y) + p(x) \quad \text{s.t.} \quad x \geq 0, \quad (3.10)$$

for some prior $p(x)$, where $NLL_{\text{Pois}}(x, y) = 1^T x - y^T \log x$. For many choices of the prior, this problem is not easy to solve directly because of its complexity. However,
separating the prior term from the negative log-likelihood term can reduce its complexity and give a simpler algorithm for solving it.

The ADMM method fits this requirement [RGE16], and can solve this problem by separating $x$ into $x$ and $v$ and constraining them to be equal, resulting in the following equivalent optimization problem:

$$\hat{x}_{MAP}, \hat{v} = \arg\min_{x,v} NLL_{Pois}(x,y) + p(v) \quad \text{s.t.} \quad x = v \geq 0,$$

and then solving it by iterating over the following three steps:

$$v_{k+1} = \arg\min_v p(v) + \frac{1}{2} \|x_k - v + \mu_k\|_2^2$$

$$x_{k+1} = \arg\min_x NLL_{Pois}(x,y) + \frac{1}{2} \|x - v_{k+1} + \mu_k\|_2^2 \quad \text{s.t.} \quad x \geq 0$$

$$\mu_{k+1} = \mu_k + x_{k+1} - v_{k+1},$$

where $\mu_0 = 0, x_0 = y$.

Equation 3.12 is the MAP optimization problem for the Gaussian denoising of $x_k + \mu_k$, and can therefore be solved using some Gaussian denoiser $D$.

The second step in each iteration involves solving a separable optimization problem, and its solution can be thought of as operating the proximal operator of the Poisson negative log-likelihood on each pixel of $v_{k+1} - \mu_k$. There is a closed form solution for this proximal operator, derived below:

$$\text{prox}_{NLL(\cdot,y)}(z) = \arg\min_x NLL_{Pois}(x,y) + \frac{1}{2} (x - z)^2 \quad \text{s.t.} \quad x \geq 0$$

$$= \arg\min_x x - y \cdot \log x + \frac{1}{2} (x - z)^2 \quad \text{s.t.} \quad x \geq 0 \quad \text{(3.15)}$$

The derivative of the optimization objective should be zero at the minimum:

$$\frac{d}{dx} \left(x - y \cdot \log x + \frac{1}{2} (x - z)^2\right) = 1 - \frac{y}{x} + x - z = 0$$

$$x^2 - (z-1)x - y = 0$$

$$x = \frac{z - 1 \pm \sqrt{(z-1)^2 + 4y}}{2}.$$ \hfill (3.16)

Since $x \geq 0$, the only solution that satisfy the constraint is:

$$\text{prox}_{NLL(\cdot,y)}(z) = \frac{z - 1 + \sqrt{(z-1)^2 + 4y}}{2},$$ \hfill (3.17)

which is a fast and simple function to use in the ADMM scheme. Overall, each iteration
of the ADMM algorithm is:

\[ v_{k+1} = D(x_k + \mu_k) \]  \hspace{1cm} (3.18)
\[ x_{k+1} = \text{prox}_{\mathcal{NLL}(\cdot,y)}(v_{k+1} - \mu_k) \]  \hspace{1cm} (3.19)
\[ \mu_{k+1} = \mu_k + x_{k+1} - v_{k+1} \]  \hspace{1cm} (3.20)

where the \( \text{prox}_{\mathcal{NLL}(\cdot,y)} \) is operating elementwise.

In this method, a Gaussian denoiser can be used for the task of Poisson denoising with no limitation on the image SNR, unlike for the VST method. The cost of this method is its requirement to use the Gaussian denoiser for each ADMM iteration, making it more computation demanding.

### 3.3 Deep Neural Networks for Poisson Denoising

Deep convolutional neural networks have shown remarkable success in many fields, including image denoising. While most image denoising architectures focus on Gaussian noise, there are also Poisson denoising neural networks. In principle, most of the networks are simply trained for the specific desired noise distribution that is to be cleaned, and the network architecture rarely reflects this noise distribution. Nevertheless, these neural network models have surpassed almost all classical Poisson denoising methods so far.

#### 3.3.1 DenoiseNet

The authors of [RLGB17] proposed a deep neural network architecture termed DenoiseNet for Gaussian and Poisson denoising of images, depicted in Figure 3.1:
This network has 650K trained parameters and was trained over the PASCAL VOC image dataset [EGW+10]. In addition to surpassing all previous classical methods for Poisson denoising of images with peak noise of 1 and above by about 0.1 to 0.3 dB, the inference time was of about 40 ms using a Titan-X GPU, an order of magnitude faster than the leading classical denoiser so far, I+VST+BM3D. However, no results are reported for noise peak values lower than 1.

Having access to labeled image dataset, the DenoiseNet model was also trained for a class-aware image denoising task – five different models were trained, each for a different class label. Using the class-aware denoisers on noisy images of their specific class, the performance was improved by an additional 0.15 to 0.30 dB, and the image quality was vastly improved.

### 3.3.2 VST-NET

Inspired by the traditional VST approach for Poisson image denoising, the authors of [ZZLW19] had incorporated it into a deep neural network architecture, which they called VST-NET, and is depicted in Figure 3.2:
The network is composed of three parts, termed SubNet1, SubNet2 and SubNet3. SubNet1 performs an Anscombe-like VST on the Poisson noisy input image, SubNet2 applies a Gaussian denoising on the resulting image, and SubNet3 implements the inversed VST to obtain the cleaned image. In principle, the network structure resembles DnCNN [ZZC+16] structure, but with 7×7 convolutional filters and two additional convolutions before and after this structure.

This network has 1.4 million parameters, and it is trained for peak values 0.1, 2, 10 and 30. The performance of the trained network on the BSD68 image dataset at peak value of 2 is about 0.5 dB lower than I+VST+BM3D, but is better on Set11 image dataset for most images.

3.3.3 MC²RNet

Another neural network architecture used for Poisson noise removal is the multi-scale cross-path concatenation residual network MC²RNet [SLZ+18]. This network, shown in Figure 3.3, contains a repeating structure termed cross-path concatenation module (C²M), which combines residual connections, dilated convolutions and concatenation of feature maps from different scales.
Figure 3.3: MC²RNet architecture.

This network contained 6 C²Ms and another initial and final units with a total of 1.55M parameters, and was trained for Poisson image denoising on several peak values, the lowest of which is 1, as well as for blind Poisson denoising.

The model showed improved PSNR results on BSD68 dataset of about 0.2 dB against DenoiseNet across several peak noise values. For peak noise value of 2, for example, the non-blind model resulted in average PSNR of 23.08 dB, which is the current state-of-the-art for this task. Nevertheless, it is unclear whether the special architecture or simply the increased number of trainable parameters are the cause for its performance.
Chapter 4

The VST-CSCNet model

The VST-Net [ZZLW19] model takes a deep convolutional network for Gaussian image denoising, such as DnCNN [ZZC+16], and uses it for Poisson image denoising using a variance stabilizing transform. While achieving good results, it has several drawbacks – for example, it is not clear why the denoiser part of the convolutional neural network should work better on an image with a Gaussian distribution of noise rather than on an image with a Poisson noise distribution, due to the lack of explainability in this part of the network. In addition, the variance stabilizing transform part of the network can be reduced to a linear convolutional layer, producing a feature map with several channels, making it very different from a traditional VST. Furthermore, the resulting PSNR values for denoising Poisson-distributed noisy images of the BSD68 test dataset of images (22.28 dB for peak value of 2) is significantly lower than that of other network architectures, such as DenoiseNet [RLGB17] (22.90 dB) and MC2RNet [SLZ+18] (23.08 dB).

As opposed to the above, we propose to use a more direct VST approach in the neural network, with solid theoretical foundations.

4.1 The Model

Our approach utilizes the CSCNet model [SE19] as a Gaussian denoiser in a regular VST scheme in a single model, trained end-to-end on Poisson noisy images. For the VST we use the Anscombe transform and the closed-form approximation of the exact unbiased inversed Anscombe transform [MF11a], implemented as elementwise operations directly on the GPU using CUDA.

The advantages of this method over VST-Net are:

- The variance stabilizing transform and its unbiased inverse transform are known and not learned. This removes some unnecessary parameters from the learning process and is fully explainable.
- The CSCNet model is inherently an MMSE estimator for the clean image under Gaussian noise, and thus fits the strategy of using a VST.
• The CSCNet model is explainable, based on a generative model for the clean image.

An illustration of our proposed network is shown in Figure 4.1, where $A$ and $A^{-1}_{\text{unbiased,approx}}$ are the Anscombe transform and the closed-form approximation to the exact unbiased inversed transform, respectively.

![Figure 4.1: The VST-CSCNet model. (a) i.i.d. Poisson distributed input image, (b) i.i.d, approximately gaussian noise, (c) cleaned transformed image, (d) cleaned image.](image.png)

### 4.2 The Dataset

For the training of the model we used the same dataset that was used for the original CSCNet training, with clean images taken from the Waterloo Exploration Dataset [MDW+17] and 432 images from BSD [AMFM11]. Each model was trained for the denoising of a specific peak value, and the noisy input images were generated by scaling the clean images to the $[0, \text{peak value}]$ range where $\text{peak value} \in \{0.1, 2\}$, taking a random $128 \times 128$ crop, applying random horizontal and vertical flips and sampling from a Poisson distribution based on the this crop. All further proposed models were trained on this data as well.

### 4.3 Model parameters

The structure of the CSC model was the same as its structure in the original paper [SE19]: The dictionaries contained $F = 175$ filters of kernel size $11 \times 11$, using a stride of 8 and 12 LISTA iterations. Overall, this model have the same number of parameters as the CSCNet model – 63.7K.

### 4.4 Training

All of the code used for this work is available at [https://github.com/assaf127/CSCPoisson](https://github.com/assaf127/CSCPoisson).

The model was trained end-to-end for peak values of 2.0 and 0.1, using $L_2$ loss with the ADAM optimizer [KB15] with $\epsilon = 10^{-3}$. For peak value of 2.0, the learning rate
was initialized at $2 \cdot 10^{-4}$ and was decreased by 0.65 every 40 epochs. For peak value of 0.1, the learning rate was had to be lowered due to divergence problems - it was initialized at $2 \cdot 10^{-5}$ and was decreased by 0.65 every 60 epochs. For both peak values the training was iterated over 250 epochs.

### 4.5 Results

The results of the VST-CSCNet model on several noisy images with peak noise of 2.0 of Set12 are compared to the results of the classical I+VST+BM3D algorithm in Figure 4.2.

![Figure 4.2: Denoising results of the VST-CSCNet model for peak = 2.0. (a) clean images, (b) Poisson noisy images, peak = 2.0, (c) I+VST+BM3D results, (d) VST-CSCNet results.](image)

In addition, the VST-CSCNet model was used also for the denoising of the BSD68 test images with peak noise of 2.0, and the average PSNR was 23.25 dB, which is 0.17 dB higher than the current state-of-the-art result of MC$^2$RNet.

The results of the model trained for peak noise value of 0.1 on the same images are shown in Figure 4.3 and compared to I+VST+BM3D.
Figure 4.3: Denoising results of the VST-CSCNet model for peak = 0.1. (a) clean images, (b) Poisson noisy images, peak = 0.1, (c) I+VST+BM3D results, (d) VST-CSCNet results.

As can be seen in the images, the model does not perform well for this very low peak value, which can be explained by the unsuccessful variance stabilization of the Anscombe transformation under this condition. The average PSNR on the BSD68 test images with noise peak of 0.1 was 18.85 dB for this model, which is 0.23 dB lower than the average PSNR obtained by I+VST+BM3D. This poor but somewhat expected result prompted us to design a better architecture for the denoising of very low peak Poisson noisy images.
Chapter 5

The ADMM-CSCNet model

Due to the failure of the VST-CSCNet model to train on the very low peak noise of 0.1, probably due to the incompatibility of the VST method to low peak values, a different, more compatible approach should be taken.

Our proposed model to handle this lower regime of peak noise values uses the Plug-and-Play Prior scheme with the CSCNet Gaussian denoiser.

5.1 The Model

Our ADMM-CSCNet model is based on the Plug-and-Play Prior scheme for Poisson denoising [RGE16]. The Gaussian denoiser $D(\cdot)$ used is the CSCNet model, and the ADMM algorithm is unrolled for a finite number of iterations $M$. Each initial offset for the strided convolutions used in the CSCNet model is treated as a separate denoiser over all the ADMM stages, and the resulting estimates for the original image are finally averaged to produce the cleaned image.

The MAP optimization problem to be solved is:

$$\hat{\Gamma}_{MAP} = \arg\min_{\Gamma} \text{NLL}_{\text{Pois}}(D\Gamma, y) + \lambda \|\Gamma\|_1 \quad \text{s.t.} \quad D\Gamma \geq 0,$$

which is equivalent to:

$$\hat{\Gamma}_{MAP}, \hat{S}_{MAP} = \arg\min_{\Gamma, S} \text{NLL}_{\text{Pois}}(S, y) + \lambda \|\Gamma\|_1 \quad \text{s.t.} \quad D\Gamma = S \geq 0.$$ (5.2)

This problem can be solved with ADMM, where each iteration includes solving:

$$\Gamma_{k+1} = \arg\min_{\Gamma} \lambda \|\Gamma\|_1 + \frac{1}{2} \|S_k - D\Gamma + \mu_k\|^2$$ (5.3)

$$S_{k+1} = \arg\min_{S} \text{NLL}_{\text{Pois}}(S, y) + \frac{1}{2} \|S - D\Gamma_{k+1} + \mu_k\|^2 \quad \text{s.t.} \quad S \geq 0$$ (5.4)

$$\mu_{k+1} = \mu_k + S_{k+1} - D\Gamma_{k+1},$$ (5.5)
and the solutions are:

\[
\begin{align*}
\Gamma_{k+1} &= CSCNet(S_k + \mu_k) \quad (5.6) \\
S_{k+1} &= \text{prox}_{\text{NLL}(. \cdot)}(D\Gamma_{k+1} - \mu_k) \quad (5.7) \\
\mu_{k+1} &= \mu_k + S_{k+1} - D\Gamma_{k+1}, \quad (5.8)
\end{align*}
\]

where the value of \( \Gamma_{k+1} \) is obtained with the LISTA process in the CSCNet model. The dictionaries \( A \) and \( B \) used in the LISTA process are shared between different iterations of the overall ADMM process, but the threshold parameters \( \tau_{i,j} \) for the soft threshold function are different for the different iterations because of the different noise levels present. The dictionary \( D \) in the above equation is the reconstruction convolutional dictionary \( C \) that appears in the CSCNet model and is also shared between all the ADMM iterations.

The learned parameters of the model are:

- The convolution dictionaries \( A, B \) and \( C \) that are used in the LISTA process and in the image reconstruction,
- The threshold parameters \( \tau_{k,i} \) for \( k \in [M] \) and \( i \in [F] \), where \( F \) is the number of filters in the convolution dictionary.

The proximal operator of the Poisson negative log-likelihood function (Equation 3.17) was implemented on the GPU with CUDA, and \( \Gamma_k \) was used as the initialization for the LISTA process in stage \( k + 1 \) for finding \( \Gamma_{k+1} \). An illustration of a single ADMM stage is shown in Figure 5.1:

![ADMM algorithm diagram](image)

Figure 5.1: A single iteration of the ADMM algorithm in the ADMM-CSCNet model.

The number of parameters of the ADMM-CSCNet model is almost the same as for the CSCNet model, with only a small number of extra threshold parameters \( \tau_{k,i} \).

### 5.2 Multiscale ADMM-CSCNet model

Treating an image across several different scales is a widely used practice in many image processing fields, and usually leads to better image qualities. This is often explained by the existence of elements of different scales in the image. In addition, as previously discussed in Section 3.2.2, the binning method is a useful way to clean very low peak
noise Poisson images. We propose to incorporate these concepts into our denoising networks for better cleaning of low SNR Poisson images.

The proposed multiscale model first uses the binning method by downscaling the noisy image, cleans it using the ADMM-CSCNet model, and upscale it to get an initial estimate \( \tilde{x} \) for the clean image \( x \).

Denoting the downsampling operator as \( \downarrow (\cdot) \), the ADMM-CSCNet model application as \( \text{ADMM-CSCNet}(\cdot) \) and the upsampling operator as \( \uparrow (\cdot) \), the first stage of the proposed model can be defined as:

\[
\tilde{x} = \uparrow \left( \text{ADMM-CSCNet}(\downarrow (x)) \right).
\] (5.9)

Next, we use the estimate \( \tilde{x} \) as a regularizer for another ADMM-CSCNet model. Since the first ADMM-CSCNet model produces an approximate MMSE estimator for the clean image, it is reasonable to add a regularizing term for a second cleaning stage, demanding the solution to be close to the estimate \( \tilde{x} \) in terms of \( L_2 \) distance (which the MMSE minimizes). This leads to the following optimization problem for the second stage:

\[
\hat{\Gamma}_{MAP}, \hat{S}_{MAP} = \arg\min_{\Gamma, S} NLL_{\text{Pois}}(S, y) + \lambda \| \Gamma \|_1 + \alpha \| D\Gamma - \tilde{x} \|_2^2
\] (5.10)

subject to \( D\Gamma = S \geq 0 \), for some \( \alpha > 0 \). This problem can be solved using the Plug-and-Play Prior scheme with ADMM in a similar manner to the single scale model, with the only difference in the first optimization equation (Equation 5.3):

\[
\Gamma_{k+1} = \arg\min_{\Gamma} \lambda \| \Gamma \|_1 + \frac{1}{2} \| S_k - D\Gamma + \mu_k \|_2^2 + \alpha \| D\Gamma - \tilde{x} \|_2^2.
\] (5.11)

This problem can be rewritten as:

\[
\Gamma_{k+1} = \arg\min_{\Gamma} \lambda \| \Gamma \|_1 + (\alpha + 0.5) \left\| D\Gamma - \left( \frac{\alpha}{\alpha + 0.5} \tilde{x} + \frac{0.5}{\alpha + 0.5} (S_k + \mu_k) \right) \right\|_2^2.
\] (5.12)

Denoting \( \alpha' = \frac{0.5}{\alpha + 0.5} \) and \( \lambda' = \frac{\lambda}{2(\alpha + 0.5)} \), this problem can be written as:

\[
\Gamma_{k+1} = \arg\min_{\Gamma} \lambda' \| \Gamma \|_1 + (\alpha + 0.5) \left\| D\Gamma - \left( (1 - \alpha') \tilde{x} + \alpha' (S_k + \mu_k) \right) \right\|_2^2,
\] (5.13)

and it can be solved using another ADMM-CSCNet model on \( (1 - \alpha') \tilde{x} + \alpha' (S_k + \mu_k) \), which is a weighted average between the previous estimate and the current image to clean.

Overall, the model is depicted in Figure 5.2:
The downscaling and upscaling operators can also be learned, and we used a dilated convolution layer for the downscaling and an interpolation followed by a convolutional layer for the upscaling.

The dictionaries $A$, $B$ and $C$ used for the first stage are different from those of the second stage, as well as the threshold parameters $\tau_{k,i}$. In addition, the $\alpha'$ parameter is also learned.

The multiscale model can also be used for more than two scales, by downscaling the noisy image several times, and using the denoising result of each lower resolution image for the denoising of a higher resolution image.

### 5.3 Training

The training was performed in a similar manner to the training of the VST-CSC model. The ADMM algorithm was enrolled $M = 5$ times, $S_0$ was initialized as $y$ or using an initial cleaning with the VST-CSC model, $\mu_0$ was initialized as 0 and $\Gamma_0$ was initialized as $B^T y$ and each $\Gamma_k$ was the initial $\Gamma$ for the LISTA iterations of $\Gamma_{k+1}$. The dictionaries contained 175 filters of kernel size $11 \times 11$ using stride of 8, and $N = 12$ LISTA iterations were performed at each ADMM stage. The model was trained for peak values of 2.0 and 0.1, using the $L_2$ loss with the ADAM optimizer [KB15] with $\epsilon = 10^{-3}$. The learning rate was initialized at $2 \cdot 10^{-4}$ and was decreased by 0.65 every 40 epochs, and the training was iterated over 250 epochs. The multiscale models were trained under the same parameters for a two-scale model and for a three-scale model, with a scale factor of 2.

The single-scale ADMM-CSCNet model used 64.4K parameters, the two-scale model used 128.9K parameters and the three-scale model used 193.2K parameters.
Another possible option that was attempted for the two-scales model was to add an $L_2$ loss term for the intermediate image estimate $\tilde{x}$ – since the estimate is obtained from an ADMM-CSCNet model, it should be an MMSE estimator for the clean image $x$, and its $L_2$ distance from it should be low. The loss term for these tests was:

$$\text{Loss}(x, \tilde{x}, \hat{x}) = (1 - \gamma) \|\hat{x} - x\|_2^2 + \gamma \|\tilde{x} - x\|_2^2,$$

where $\gamma \in [0, 1)$ and $\hat{x}$ is the final output of the model.

### 5.4 Results

The results of the single-scale ADMM-CSCNet models on several noisy images with peak noise of 2.0 of Set12 are compared to the results of the VST-CSCNet model and the classical I+VST+BM3D algorithm in Figure 5.3:

![Figure 5.3: Single scale ADMM-CSCNet denoising results for peak noise = 2.0. (a) clean images, (b) Poisson noisy images, peak noise = 2.0, (c) I+VST+BM3D results, (d) VST-CSCNet results, (e) single-scale ADMM-CSCNet results.](image)

In addition, the single-scale ADMM-CSCNet model was also tested on the BSD68 test images with peak noise of 2.0, and the average PSNR was 23.29 dB which is 0.04 dB higher than the VST-CSCNet, indicating that perhaps for this peak noise value the VST model is enough.

The ADMM-CSCNet model was also trained for the very low peak noise value of
0.1, and its denoising results on several Set12 images are compared to I+VST+BM3D and to VST-CSCNet in Figure 5.4:

As depicted in Figure 5.4, the ADMM-CSCNet model afforded much better results than the VST-CSCNet model under this very low peak value. In addition, it produced images with less artifacts than the I+VST+BM3D method, with sharper edges and more details for this hard denoising task. The average PSNR for BSD68 images was 19.32 dB for this model, which is 0.24 dB higher than for I+VST+BM3D and 0.47 dB higher than for VST-CSCNet. The multiscale model gave very similar results for the single-scale model in terms of PSNR and visual image quality for most images, but in some cases for the very low peak value of 0.1 it gave smoother images while preserving the edges, as can be seen in Figure 5.5.
Figure 5.5: Single scale ADMM-CSCNet denoising results for peak noise = 0.1. (a) clean images, (b) Poisson noisy images, peak noise = 0.1, (c) I+VST+BM3D results, (d) single-scale ADMM-CSCNet results, (e) two-scale ADMM-CSCNet results, (d) three-scale ADMM-CSCNet results.

The average PSNR results of the trained models on BSD68 image dataset are presented and compared to other Poisson denoising methods in Table 5.1.

Table 5.1: Denoising performance (PSNR, dB) on the BSD68 dataset.

<table>
<thead>
<tr>
<th>Method</th>
<th>Peak noise = 2.0</th>
<th>Peak noise = 0.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>I+VST+BM3D$^a$</td>
<td>22.59</td>
<td>19.08</td>
</tr>
<tr>
<td>DenoiseNet$^b$</td>
<td>22.90</td>
<td>-</td>
</tr>
<tr>
<td>VST-NET$^c$</td>
<td>22.28</td>
<td>-</td>
</tr>
<tr>
<td>MC$^2$RNet$^d$</td>
<td>23.08</td>
<td>-</td>
</tr>
<tr>
<td>VST-CSCNet</td>
<td>23.25</td>
<td>18.85</td>
</tr>
<tr>
<td>ADMM-CSCNet$_1$</td>
<td>23.29</td>
<td>19.32</td>
</tr>
<tr>
<td>ADMM-CSCNet$_2$</td>
<td><strong>23.34</strong></td>
<td><strong>19.36</strong></td>
</tr>
<tr>
<td>ADMM-CSCNet$_3$</td>
<td>23.31</td>
<td>19.36</td>
</tr>
</tbody>
</table>

In **bold** the best results for a given peak noise value.

a) [AF16], b) [RLGB17], c) [ZZLW19], d) [SLZ]+18.

The trained models were also tested on some standard test images (shown in Figure 5.6), and the PSNR results were averaged over 15 noise realizations and compared to other methods in Table 5.2.

35
Figure 5.6: Standard test images. From left to right: Saturn, Monarch, Barbara, House, Cameraman.

Table 5.2: Denoising performance (PSNR, dB) on standard images.

<table>
<thead>
<tr>
<th>Method</th>
<th>Peak</th>
<th>Saturn</th>
<th>Monarch</th>
<th>Barbara</th>
<th>House</th>
<th>Cameraman</th>
</tr>
</thead>
<tbody>
<tr>
<td>Poisson NLM</td>
<td>17.22</td>
<td>13.62</td>
<td>14.08</td>
<td>14.28</td>
<td>13.92</td>
<td></td>
</tr>
<tr>
<td>Poisson NLSPCA</td>
<td>19.23</td>
<td>14.47</td>
<td>16.45</td>
<td>16.80</td>
<td>15.76</td>
<td></td>
</tr>
<tr>
<td>I+VST+BM3D</td>
<td>20.53</td>
<td>15.34</td>
<td>17.99</td>
<td>19.07</td>
<td>17.54</td>
<td></td>
</tr>
<tr>
<td>VST-NET$^d$</td>
<td>21.67</td>
<td>15.13</td>
<td>18.00</td>
<td>19.30</td>
<td>17.42</td>
<td></td>
</tr>
<tr>
<td>VST-CSCNet</td>
<td>20.90</td>
<td>15.16</td>
<td>18.01</td>
<td>19.26</td>
<td>18.13</td>
<td></td>
</tr>
<tr>
<td>ADMM-CSCNet$_1$</td>
<td>22.15</td>
<td>15.70</td>
<td>18.00</td>
<td>19.25</td>
<td>18.11</td>
<td></td>
</tr>
<tr>
<td>ADMM-CSCNet$_2$</td>
<td>22.32</td>
<td>15.88</td>
<td>18.01</td>
<td>19.26</td>
<td>18.13</td>
<td></td>
</tr>
<tr>
<td>ADMM-CSCNet$_3$</td>
<td>22.73</td>
<td>15.90</td>
<td>18.02</td>
<td>19.29</td>
<td>18.09</td>
<td></td>
</tr>
<tr>
<td>Poisson NLM</td>
<td>27.76</td>
<td>20.45</td>
<td>21.52</td>
<td>23.43</td>
<td>13.92</td>
<td></td>
</tr>
<tr>
<td>Poisson NLSPCA</td>
<td>27.68</td>
<td>19.67</td>
<td>21.69</td>
<td>24.09</td>
<td>20.65</td>
<td></td>
</tr>
<tr>
<td>I+VST+BM3D</td>
<td>29.07</td>
<td>21.35</td>
<td>22.09</td>
<td>25.16</td>
<td>22.42</td>
<td></td>
</tr>
<tr>
<td>VST-NET$^2$</td>
<td>28.77</td>
<td>22.01</td>
<td>22.11</td>
<td>24.91</td>
<td>22.27</td>
<td></td>
</tr>
<tr>
<td>VST-CSCNet</td>
<td>29.12</td>
<td>22.21</td>
<td>21.93</td>
<td>25.55</td>
<td>23.26</td>
<td></td>
</tr>
<tr>
<td>ADMM-CSCNet$_1$</td>
<td>29.39</td>
<td>22.29</td>
<td>21.93</td>
<td>25.72</td>
<td>23.30</td>
<td></td>
</tr>
<tr>
<td>ADMM-CSCNet$_2$</td>
<td>29.55</td>
<td>22.40</td>
<td>22.05</td>
<td>25.87</td>
<td>23.46</td>
<td></td>
</tr>
<tr>
<td>ADMM-CSCNet$_3$</td>
<td>29.27</td>
<td>22.32</td>
<td>22.01</td>
<td>25.73</td>
<td>23.32</td>
<td></td>
</tr>
</tbody>
</table>

Averaged PSNR results for 15 noise realizations of each image are reported for our model. In **bold** the best results for a given peak noise value.

a) [DTD10], b) [SHDW14], c) [AF16], d) [ZZLW19].

As can be seen in these results, the ADMM-CSCNet models give better PSNR results than the VST-CSCNet method and all previous classical and deep learning methods. The multiscale model gives slightly better results than the single-scale model, especially for low SNR images, and produces smoother images. Our models do not show improvement over other methods on the Barbara image, perhaps because this image contains some repeating patterns that can benefit from including a self-similarity prior, which our models do not use.

In addition, the $\gamma$ parameter for the $L_2$ loss term for the intermediate image estimate $\tilde{x}$ was taken from $\{0, 10^{-3}, 10^{-2}, 10^{-1}\}$, and no effect on the final PSNR for the BSD68 dataset was observed. This can be explained by the fact that the model already has to bring $\tilde{x}$ close to the clean image $x$ in order for the added regularization term to help in the cleaning process, so for the optimized model this added term should also be at a minimum.

36
Chapter 6

Conclusion

This work proposes several novel deep convolutional neural network architectures for the removal of Poisson noise from images. These architectures are all based on classical algorithms, and their operation and weights can therefore be explained. Our models include the Gaussian denoising network structure of CSCNet [SE19], which is a successful, explainable deep convolutional neural network that relies on sparse representation principles. Leveraging this Gaussian denoiser, our network architectures can clean Poisson noisy images.

Our first model, the VST-CSCNet, uses the Anscombe as a variance stabilizing transform (VST) to give an approximately Gaussian noisy image, which is cleaned with the CSCNet model and transformed back with an inverse transform to give the clean image. This relatively simple model is trained end-to-end for Poisson denoising, and produces state-of-the-art results in the denoising of images with peak value of 2, despite having far less parameters than other leading deep learning methods. However, the Anscombe VST fails at very low peak values, and this method does not work as well for the very low peak value of 0.1.

Our next model, ADMM-CSCNet, uses the Plug-and-Play Prior method [RGE16] for Poisson denoising, by applying the CSCNet model as a Gaussian denoiser inside an ADMM algorithm. This architecture gave slightly better results than the VST-CSCNet model for peak value of 2.0, and managed to clean images successfully at the very low peak value of 0.1 where VST-CSCNet failed. In addition, multiscale elements were incorporated to this network structure, which resulted in slightly better denoising results and image quality, especially for very low peak values.

While having some successful results, our models could perhaps perform better by relying on self-similarities in the images to be cleaned. This prior is very helpful in low SNR images where very little data is available and is extensively used in other works for low peak Poisson denoising ([GE14], [SHDW14], [DTD10], [AF16]).

Another possible future direction can be modifying our proposed networks to solve more complex inverse problems, such as deblurring, inpainting, super resolution and more. In addition, cleaning real noise is also an interesting task to handle – learning the
VST or the proximal operator can generalize our models for many different real noise types.


The architectures, the models studied in this work, present multiple convolutional layers with a goal to extract information from the images, based on classical image processing techniques. The models are

\[ \text{CSCNet} \]

and the classic Gaussian filter.

The model architecture is shown in the following schematic diagram. The model is trained on the \text{CSCNet} model, which is a convolutional sparse coding model. It is trained on the images with noise, and finally, the \text{CSCNet} model is used.

The model achieves state-of-the-art results for the standard \text{BSD68} dataset, and reaches results for low noise images. In addition, the model architecture is used for different image processing techniques.

Moreover, it is shown that with the classical image processing techniques, the \text{Anscombe} model is trained on the images with noise, and finally, the \text{CSCNet} model is used.
ניקיית רעש תמונות היא משימה ייעודית באופן&_x000D_&_x000D_irosית וחשובה. בכל תמונה קיימ רעש מספרים שלhtarfatוג.oauthית נקה הבוקע על פי שלטים ילוס וה谄ונה והצטלבות עצמה, ויושלחחצב הסמארת או בחלקי יניקי הרעש. התוצאותمشارיימם, כל לבוש בריאית אחר, יפים וה/screenות פועלשם pzא룽מוות הflammתה ועמהית תחת תחנת האורח התולעה, כל פיקסל של חיות emולמצאה מה שמיומן יותר. קבלת תמונה נקה מrequestCodeים היא משימה שמחבה ומשמעת עתה решение פוטווכי כל פיקסל של חיישן המצלמה בдесь בלא, ונוח לא מתלע את הורשה באימוץ הflammתה פואסוני, והספק לקרא פוטווכי כדי ומשמעי באימוץ הflammתה פואסוני, ובתרחישיםvlaとしても, ונוח או פרך באימוץ הflammתה פואסוני, ומשמעי יניקי פואסוני בלא, נוח או פרך הוא פואסוני, ובתרחישיםvlaとしても, ונוח או פרך הוא פואסוני, ובתרחישיםvlaとしても, ונוח או פרך הוא פואסוני, ובתרחישיםvlaとしても, ונוח או פרך הוא פואסוני, ובתרחישיםvlaとしても, ונוח או פרך הוא פואסוני, ובתרחישיםvlaとしても, ונוח או פרך הוא פואסוני, ובתרחישיםvlaとしても, ונוח או פרך הוא פואסוני, ובתרחישיםvlaとしても, ונוח או פרך הוא פואסוני, ובתרחישיםvlaとしても, ונוח או פרך הוא פואסוני, ובתרחישיםvlaとしても, ונוח או פרך הוא פואסוני, ובתרחישיםvlaとしても, ונוח או פרך הוא פואסוני, ובתרחישיםvlaとしても, ונוח או פרך הוא פואסוני, ובתרחישיםvlaとしても, ונוח או פרך הוא פואסוני, ובתרחישיםvlaとしても, ונוח או פרך הוא פואסוני, ובתרחישיםvlaとしても, ונוח או פרך הוא פואסוני, ובתרחישיםvlaとしても, ונוח או פרך הוא פואסוני, ובתרחישיםvlaとしても, ונוח או פרך הוא פואסוני, ובתרחישיםvlaとしても, ונוח או פרך הוא פואסוני, ובתרחישיםvlaとしても, ונוח או פרך הוא פואסוני, ובתרחישיםvlaとしても, ונוח או פרך הוא פואסוני, ובתרחישיםvlaとしても, ונוח או פרך הוא פואסוני, ובתרחישיםvlaとしても, ונוח או פרך הוא פואסוני, ובתרחישיםvlaとしても, ונוח או פרך הוא פואסוני, ובתרחישיםvlaとしても, ונוח או 프
המחקר בוצע בהנחייה של פרופסור מיכאל על על פי חוקת המחלקה לתמרון מחודש והrias.

תודה

אני מודה לפורפ, מיכאל על על תמיכת הרבח במחזור המחקר.

عجبת מחקר ומקודשת לאשתי.

הכרת תודה מוטורה לעצמי על מיומנויות מחקר זה.
ניקי רעשה פואסוני מתמודד
באימוץ ולהת揚 נוירונים בחשראת
אלגוריתמים מבוססי מילון קלאסים

hibor על מחקר

לשם מילוי כרキー של הדרישות לקבלת התואר
מוניטור ממ Erotic ומנוע מבודי המחבר

אסף מעודה

הוגש לפנינו הטכניון — מכון טכנולוגי לישראל
תחום ה"iterals" בחיפה אוקטובר 2021
ניקויOURSE פואטים ממוחות
באמ-gunגוע רשתות ניירות בהשראת
אלגוריתמים מתוכננים Miול קלאסיים

אסף מעודה