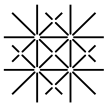


University of Basel

Graph Denoising for Molecular Imaging

Master's Thesis

Natural Science Faculty of the University of Basel
Department of Mathematics and Computer Science
Signals and Data group

Examiner: Prof. Dr. Ivan Dokmanić
Supervisor: Dr. Valentin Debarnot

Cédric Mendelin
cedric.mendelin@stud.unibas.ch
2014-469-274

12.06.2022

Acknowledgments

I would like to express my deepest appreciation to Prof. Dr. Ivan Dokmanić for giving me the opportunity to conduct my Master's Thesis in the Signals and Data group.

Further, I am extremely grateful for the support by my supervisors Dr. Valentin Debarnot and Cheng Shi. We had inspiring discussions throughout the project and their valuable feedback was more than welcome, helping me to finish my Thesis in an appropriate quality.

Many thanks to Patrick Arnold and Marc Hennemann for proofreading this report, their feedback was greatly appreciated.

Words cannot express my gratitude for my wife Luana, my family and friends, for the support during the last years. This accomplishment would not have been possible without them.

Abstract

Cryo-Electron Microscopy (cryo-EM) and Computer Tomography (CT) are two molecular imaging methods. In both scenarios, observations are collected from a biological sample. During reconstruction, the aim is, to approximate the underlying biological sample from observations. For CT, the reconstruction can be in 2D or 3D, for cryo-EM it is in 3D. Cryo-EM enables the view of molecules in near-atomic resolution and therefore got a lot of attention in recent years. Due to ground-breaking improvements regarding hardware and data processing, the field of research has highly improved. Reconstruction is challenging due to the present of enormous noise and unknown projection angles in the observations. CT is a similar to cryo-EM, but reconstruction is slightly easier as the problem can be in 2D and observation angles are known. Therefore, CT is well suited to implement an algorithm toward an extended version working for cryo-EM as well.

I introduce GAT-Denoiser, a Graph Neural Network, which combines Graph Attention Network with convolution and an end-to-end learning approach. It enables to denoise observations and boosts reconstruction quality with the assumption of known projection angles. During training an end-to-end approach is used where reconstruction quality is compared and not only denoised observation quality. GAT-Denoiser have been evaluated on the LoDoPaB-CT [14] dataset. It could outperform baseline algorithm BM3D measure by the reconstruction signal-to-noise-ratio (SNR). An improvement by 379.9%, 126.0% 57.7%, 27.6% for SNR_y -15 dB, -10 dB, -5 dB and 0 dB respectively could be fulfilled.

Table of Contents

Acknowledgments **ii**

Abstract **iii**

1 Notes **1**
1.1 ToDo . . . 1
1.2 Wording . . . 1
1.3 Feedback: . . . 1

2 Introduction **2**

3 Notation **4**
3.1 Molecular Imaging Methods . . . 4
3.2 Graph . . . 4
3.3 Maths . . . 5

4 Molecular Imaging Methods **6**
4.1 Computed tomography . . . 6
4.2 Cryo-EM . . . 8
4.3 Abstraction . . . 10

5 Manifolds and Graphs **12**
5.1 Graph Foundations . . . 12
5.2 Graph Laplacian & Manifolds . . . 13
5.2.1 Graph Laplacian . . . 13
5.2.2 Graph Laplacian Embedding for CT and cryo-EM . . . 13
5.3 Graph Denoising . . . 15
5.4 Graph Deep Learning . . . 16

6 GAT-Denoiser **17**
6.1 Pipeline . . . 17
6.2 Layers . . . 18
6.3 Training . . . 19

7 Results **21**

7.1 Dataset . . . 21
7.2 Project setup . . . 21
7.3 Small scale GAT-Denoiser experiments . . . 22
7.3.1 Baseline . . . 22
7.3.2 K-NN Experiments . . . 23
7.3.3 GAT Experiments . . . 25
7.3.4 Convolution Experiments . . . 26
7.3.5 Loss Experiments . . . 27
7.3.6 GAT-Denoiser Component Experiments . . . 28
7.4 Large scale Experiments . . . 29
7.4.1 Baseline . . . 29
7.4.2 GAT-Denoiser components . . . 30
7.4.3 Best Models . . . 30

8 Conclusion and Future Work **32**
8.1 Conclusion . . . 32
8.2 Future Work . . . 32

Bibliography **34**

Appendix A Mathematical Tools **37**

Appendix B Graph Laplacian Embedding **39**

Appendix C Neural Networks **43**

Appendix D Additional results **46**

1 Notes

1.1 ToDo

- Declaration on Scientific Integrity
- Link to paper GL tomography from unknown projections

1.2 Wording

- Where nur für ort angaben
- Mehrzahl/Einzahl
- Short sentences, not too nested

1.3 Feedback:

- Imaging: Maybe first abstraction, then CT and cryo-EM.
- Check Imaging Math once more. Use SO(p) for angles.

2
Introduction

Inverse Problems aim to estimate an original signal that went through a system, based on potentially noisy output signal observations. They are widely used throughout different science directions, such as Machine Learning (ML), Signal Processing, Computer Vision, Natural Language Processing and others. ML is one tool to model and solve inverse problems.

Cryo-Electron Microscopy (cryo-EM) is a molecular imaging method and gained a lot of attention in recent years. Molecules are frozen and imaged through an electron microscope. Due to ground-breaking improvements regarding hardware and data processing, the field of research has highly improved. In 2017, pioneers in the field of cryo-EM got the Nobel Prize in Chemistry[1]. Today, using cryo-EM, molecular structures can be observed with near-atomic resolution. The big challenge with cryo-EM is enormous noise and unknown observation angles.
Computed tomography (CT) is similar to cryo-EM, but reconstruction is slightly easier as the problem is in 2D and observation angles are known.
The overall goal of this Thesis is to introduce an algorithm that works with CT, but can conceptually be extended to work in 3D, thus for cryo-EM. Additionally, the focus is on the high noise domain, as current available reconstruction algorithms start to fail when dealing with too much noise. The signal-to-noise-ratio of interest is the interval between $[-15, 0]$.

In recent years, graphs got a lot of attention in ML and Graph Machine Learning is one of the most promising research areas. Graphs are a well suited data structure, simple but with high expressiveness. Especially data for which single data points tend to have a relation to other data points, graphs with its nodes and vertices are the perfect tool to capture these relationships. Data is in a graph structure already, like social networks, or they can be artificially constructed for arbitrary datasets. Besides, for some scenarios, ordinary ML algorithms fail, but Graph ML approaches have great success, e.g. dimensionality reduction for high-dimensional data.

[1] https://www.nobelprize.org/prizes/chemistry/2017/press-release/

As a result of this Thesis, a Graph Neural Network architecture is proposed, which is called *GAT-Denoiser*. GAT-Denoiser aims to denoise noisy observations to improve overall reconstruction quality. The assumption of known observations angles is defined. In the GNN architecture, convolution and Graph Attention Network (GAT) is used to denoise observations. In addition, an end-to-end learning approach is used, where the reconstruction quality is compared in the loss and not only the denoised observation quality.
GAT-Denoiser was evaluated on the LoDoPaB-CT dataset [14]. I could show that all three components contribute to learning the best GAT-Denoiser model. Further, U-Net[16] was used to further improve CT reconstruction quality. GAT-Denoiser outperformed BM3D [7] as well as U-Net [16], where observation signal-to-noise-ration (SNR) is between 0 dB and -15 dB. Compared to the best performing baseline BM3D, GAT-Denoiser could improve reconstruction SNR by 379.9% for observation SNR -15 dB. The best GAT-Denoiser models could be established, when first, GAT-Denoiser is trained with a fixed U-Net model. After some learning, in a second step, GAT-Denoiser and U-Net have been trained jointly.

The report is structured as follows:
First, notation applied throughout this Thesis is introduced in Chapter 3. Chapter 4 presents the two molecular imaging methods CT and cryo-EM and a mathematical abstraction for observation and reconstruction is introduced. Further, Chapter 5 is dedicated to manifolds and graphs, and how a meaningful embedding for CT and cryo-EM can be established. The main contribution is presented in Chapter 6, where GAT-Denoiser is introduced and results are presented in Chapter 7. Finally, conclusion and future work are presented in Chapter 8.

3
Notation

In this chapter, some basic terms are explained and their notation in this report is defined.

3.1 Molecular Imaging Methods

In molecular imaging methods CT and cryo-EM, observations are collected from a biological sample. These observations are noisy. Reconstruction from noisy observations is desired, to approximate the biological sample. Throughout this Thesis, notation p for noiseless observation and y for observation with noise is used. In practice, p is not observable directly and observed signal y needs to be denoised. Further, x is used for the biological sample and x' defines approximation by the reconstruction algorithms. In addition, N is used for the number of observations and M as the observation dimension. Consequently, $y_i \in \mathbb{R}^M$ determines i-th observation and $y_i[j] \in \mathbb{R}$ the j-th element of y_i, with $1 \leq i \leq N$ and $1 \leq j \leq M$. The same is true for p with p_i and $p_i[j]$.

Signal-to-noise-ratio Signal-to-noise-ratio (SNR) is a measure used in Signal Processing. It compares the power of an input signal to the power of noise. It is typically given in decibel (dB), with $\text{SNR}_{dB} = 10 \log_{10} \left(\frac{P_{signal}}{P_{noise}} \right)$. An SNR of smaller than 0 dB indicates more noise than signal.
From a given clean image and its noisy version, SNR can be determined. During this Thesis, the term SNR is used in two scenarios. First, to indicate how much noise is present in y. Second, as quality measure for reconstruction. To make notation clear, SNR_y is used to express the level of noise in y, where SNR is computed from y and p. Contrary, SNR refers to SNR computed from x' and x.

3.2 Graph

A graph is defined as $G = \langle V, E \rangle$, where V is a set of vertices (or nodes) and E is a set of edges (or links). Edges are defined as a set of tuples (i, j), where i and j determine the index of vertices in the graph.

Graph properties: A graph can be either *directed* or *undirected*. In a directed graph, an edge connects explicitly from one node to another, consequently edge $(i, j) \neq (j, i)$. In an undirected graph, edges have no direction and ordering does not matter, therefore $(i, j) = (j, i)$. Throughout this Thesis, undirected graphs are considered and when writing from a graph, is refers to an undirected graph.
The *neighborhood*, denoted by $\mathcal{N}(i)$, of a node i is defined as all adjacent nodes. In other words, there is an edge between neighborhood nodes and i. Further, edges can have *weights*, which is a method to define the importance of neighbors, resulting in a *weighted* graph. *Degree* of a node are the number of incoming edges.

Adjacency Matrix: The (binary) adjacency matrix of graph $G = \langle V, E \rangle$ is defined as follows:

$$A_{ij} = \begin{cases} 1 & \text{if } (i, j) \in E \\ 0, & \text{otherwise} \end{cases} \tag{3.1}$$

Matrix A has dimension $\mathbb{R}^{N \times N}$ with N as number of nodes and indices of A correspond to nodes in V. If there exists an edge between two nodes, entry in A will be set to 1, otherwise to 0. This leads to an unweighted graph, as weights of all edges will be 1. When G is undirected, corresponding adjacency matrix will be symmetric. Eigenvalues of A are called *spectrum* of G.

Graph construction with k-NN: K-nearest-neighbors (k-NN) is a graph construction algorithm. The distance between nodes is calculated with a distance measure (e.g. Euclidean distance) and for every node, $\mathcal{N}_i$ is defined as k nodes with the smallest similarity measure. During this Thesis, when writing from k it refers to k-NN parameter k.

Erdős–Rényi graph TODO: Introduce shorty

3.3 Maths

Convolution: During this Thesis, symbol $\star$ is used for the convolution operator.

Concatenation: $\|$ is used for the concatenation operator, which combines vectors end-to-end. For vectors $x = (x_0, \ldots, x_n)$ and $y = (y_0, \ldots, y_n)$ concatenation is defined as $x \| y = z$ with $z = (x_0, \ldots, x_n, y_0, \ldots, y_n)$. The operator can be applied to an arbitrary number of vectors. Consider N vectors $v_1, \ldots, v_n$, concatenation is written as $\|_{i=1}^{N} v_i$.

4

Molecular Imaging Methods

This chapter introduces two molecular imaging methods, *Computed Tomography* (CT) and *Cryo-Electron Microscopy* (cryo-EM). Further, their observation model is defined in a mathematic way and reconstruction is presented. Application of cryo-EM is a major motivation for this Thesis, as the problem is not easy to solve due to dealing with enormous noise and unknown observation angles. Moreover, cryo-EM can be seen as a problem in 3D, as original object is in 3D. CT is similar to cryo-EM, but reconstruction is slightly simpler, since it is potentially in 2D and observation angles are known. That's why it is well suited as a first step towards a cryo-EM algorithm.

4.1 Computed tomography

CT is a well established molecular imaging method. Using X-ray source, fan shaped beams are produced which scan the biological sample. Through scanning over straight lines many observations are collected, and the biological sample can be reconstructed.[2]

2D tomographic observation: Mathematically, CT observations are defined as follows:

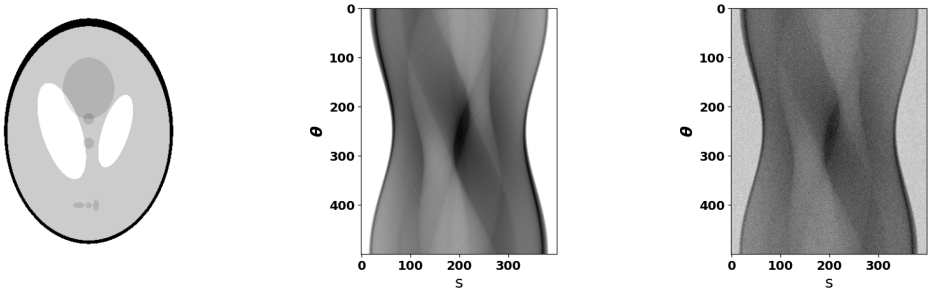
$$\begin{aligned} y_i[j] &= p_i + \eta_i[j], && \text{with } 1 \leq i \leq N \text{ and } 1 \leq j \leq M \\ &= R(x, \theta_i, s_j) + \eta_i[j], && \text{with } 1 \leq i \leq N \text{ and } 1 \leq j \leq M \end{aligned} \tag{4.1}$$

with

- $x \in L^2(\Omega)$: original object with $\Omega \subset \mathbb{R}^2$ and L^2 Lebesgue space
- $R(\cdot; \theta_i, s_j) : L^2(\Omega) \to \mathbb{R}^M, x \mapsto R(x; \theta_i, s_j)$: Radon Transform[3] with, $\theta_i \in \mathbb{S}^1$: observation angle and $s_j \in \mathbb{R}$: sampling point
- $\eta_i \in \mathbb{R}^M$: i.i.d Gaussian noise with $\eta_i[j] \sim \mathcal{N}(0, \sigma^2) \in \mathbb{R}$

[2] For further details [2]
[3] For additional information [20]

Observation illustration: Output of Radon Transform is called a *sinogram*, which is the CT observation. In Figure 4.1(a) the Shepp-Logan phantom is illustrated. It is often used as an image for simulating a brain CT and in refers to the biological sample x. Further, in Figure 4.1(b) and Figure 4.1(c) observation sinograms can be seen with and without noise respectively. θ and s must be defined to apply Radon Transform. For this example, $\theta \in \mathbb{R}^{500}$ was evenly spaced between $[0, 2\pi]$ and $dim(s) = 400$. Thus, $p \in \mathbb{R}^{500\times400}$ and can be plotted as image with resolution 500x400. Further, noise was added to reach an SNR of 10 dB.



(a) Shepp-Logan phantom : x

(b) Clean sinogram: $R(x, \theta, s)$

(c) noisy sinogram: $R(x, \theta, s) + \eta = y + \eta$

Figure 4.1: Shepp-Logan phantom and corresponding sinograms.

Tomography reconstruction: Tomographic reconstruction is a popular inverse problem. The aim is to reconstruct a biological sample based on observations. When original object is in 2D, observations are available in 1D. The reconstruction is possible for 3D objects as well, where observations are in 2D[4]. In this Thesis I focus on the 2D case of CT, which is called *classical tomography*

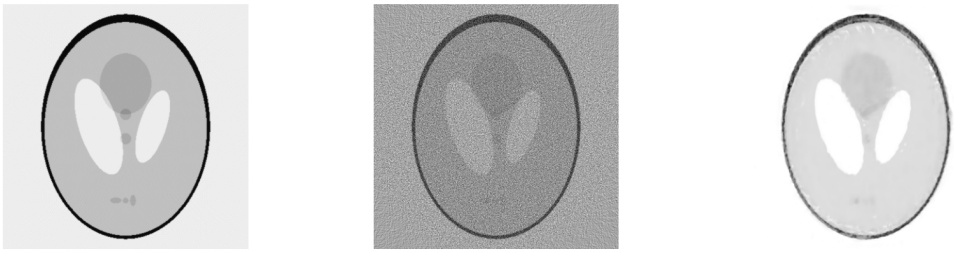
Filter Backprojection: Filter Backprojection (FBP) [5] is a reconstruction method used in classical tomography. Until recently, it was the primary method for reconstruction as it allows to inverse the Radon Transform and enables reconstruction of the original object x.
FBP can be defined as:

$$FBP(\cdot; \theta_i, s_j) : \mathbb{R}^M \to L^2(\Omega), y \mapsto FBP(y; \theta_i, s_j) \tag{4.2}$$

Where θ are the projection angles and s are the sampling points. The algorithm fails when working with highly noisy data [17], as it is not possible to draw meaningful connections anymore, noise is dominating information in the data.
Therefore, alternatives for the high noise domain are studied in recent year. Lately, neural network approaches emerged, which further process the output of FBP to increase reconstruction quality. This is the approach I will follow in this Thesis. Leuschner et al. [15] compared different Deep-Learning reconstruction methods for CT.

[4] For additional information [5]

U-Net Today's state-of-the-art reconstruction algorithms are Deep-Learning based. U-Net [16], a convolution neural network approach, performed much better compared to FBP in the comparison of Leuschner et al. [15]. Therefore, it is an interesting baseline and additionally can be combined with other ideas.
In Figure 4.2 reconstructions from Shepp-Logan phantom are presented. Noise in y is defined to reach an SNR_y of 10 dB.
The quality of noisy reconstruction is rather low, some important details are missing, and the noise dominates reconstruction. If more noise is present in y, reconstruction will be of even lower quality. Further, Figure 4.2(c) shows reconstruction with U-Net where some details are missing as well, although overall noise is drastically decreased.



(a) Clean reconstruction: $FBP(p, \theta, s)$ (b) Noisy reconstruction: $FBP(y, \theta, s)$ (c) Noisy reconstruction: $UNet(FBP(y, \theta, s))$

Figure 4.2: Shepp-Logan reconstructions with SNR_y 10 dB.

4.2 Cryo-EM

Cryo-EM is another molecular imaging method, that enables the view of molecules in near-atomic resolution. In this Thesis, for simplicity, only single-particle cryo-EM [8] is considered. When writing about cryo-EM it always refers to single-particle cryo-EM.
During the imaging process molecules are frozen in a thin layer of ice, where they are randomly oriented and positioned. Random orientation and positioning makes reconstruction challenging, but freezing allows observation of their conformation in a stable state where molecules are not moving. With an electron microscope, 2D tomographic projection images of molecules are observed, which are called *micrograph*. Frozen molecules are fragile and electron microscope needs to work with very low power (electron dose), resulting in highly noisy observations. The resulting SNR is typically smaller than 0 dB, which indicates that there is more noise than signal [17].
In addition, observed molecules are not equal in the sense that there are some structural varieties between molecules (isotopes). While observing the same molecule in ice many times, single observations could be from different isotopes.
TODO: some inconsistency with M and the number of observation. I think Delta should be Omega power M , and then indices (j,k) are taken from Delta.

3D cryo-EM observation: Mathematically, observation is defined as follows:

$$\begin{aligned} y_i &= p_i + \eta_i, && \text{with } 1 \leq i \leq N, \\ y_i &= \Pi_z(\ Rot(\ x; \theta_i)) + \eta_i, && \text{with } 1 \leq i \leq N, \end{aligned} \tag{4.3}$$

where

- $x \in L^2(\Omega)$: original object with $\Omega \subset \mathbb{R}^3$ and L^2: Lebesgue space
- $\Pi_z : L^2(\Omega) \to L^2(\tilde{\Omega}), x \mapsto \int x(\cdot, \cdot, z)dz$: z-axis projection operator, with $\tilde{\Omega} \subset \mathbb{R}^2$
- $\theta_i = [\theta_i^{(1)}, \theta_i^{(2)}, \theta_i^{(3)}]$: 3D rotation matrix with $\theta_i^{(1)}, \theta_i^{(2)}, \theta_i^{(3)} \in \mathbb{R}$ and $R_{\theta_i} = R_{e_x}(\theta_i^{(1)})R_{e_y}(\theta_i^{(2)})R_{e_z}(\theta_i^{(3)}) = [R^1_{\theta_i}, R^2_{\theta_i}, R^3_{\theta_i}] \in SO(3)$ [5]
- $Rot : L^2(\Omega) \to L^2(\Omega), Rot(x, \theta_i) = \big((x_1, x_2, x_3) \mapsto x(x_1 R^1_{\theta_i}, x_2 R^2_{\theta_i}, x_3 R^3_{\theta_i})\big)$: rotation operator
- $\eta_i \in \mathbb{R}^M$: Gaussian noise with $\eta_i[j] \sim \mathcal{N}(0, \sigma^2) \in \mathbb{R}$

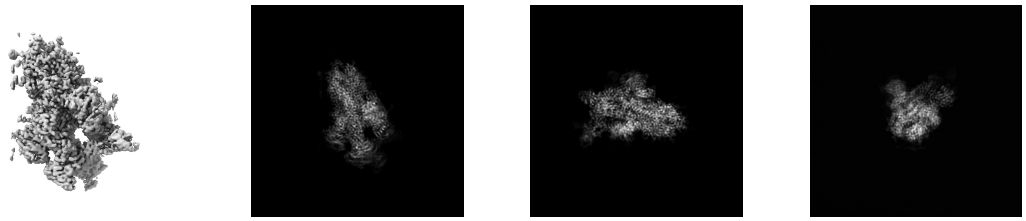
Equation 4.3 is a simplified version of cryo-EM. First, point spread function (PSF) of the microscope is not taken into account. Second, structural variety is ignored, the underlying object x is not the same for every observation. Precisely, x can be seen as a random signal from an unknown distribution defined over all possible molecules structures. In this Thesis, only the simplified version in Equation 4.3 is considered.
Morevoer, as y_i is not observable directly, discretization is needed:

$$\begin{aligned} y_i &= (\Pi_z(\ Rot(\ x; \theta_i)) + \eta_i)\,(\Delta) && , \text{ with } 1 \leq i \leq N \\ y_i[j,k] &= \Pi_z(\ Rot(\ x; \theta_i))_{j,k} + \eta_i[j,k] && , \text{ with } 1 \leq i \leq N \text{ and } 1 \leq j,k \leq M \end{aligned} \tag{4.4}$$

with

- $\Delta \subset \tilde{\Omega}^M$: sampling grid with dimension M
- $y[j,k]$, $\eta[j,k]$ and $\Pi_z(\cdot)_{j,k} \in \mathbb{R}$ with j, k as indices of the sampling grid.

Observation illustration: In Figure 4.3 some cryo-EM observations are illustrated as well as the reconstructed biological sample.



(a) COVID-19 Omicron spike (b) Observation along x-axis (c) Observation along y-axis (d) Observation along z-axis

Figure 4.3: Cryo-EM reconstruction and clean projections of COVID-19 Omicron spike [6]

[5] (for further details see A.1)

3D cryo-EM reconstruction: Cryo-EM reconstruction is defined as estimation of a 3D object from 2D observations. It can be seen as a 3D problem as the original object $x \in L^2(\Omega)$ to be reconstructed is in 3D. Based on many observed micrographs the original object x is estimated. Cryo-EM reconstruction is computational intensive and multiple steps are needed to get from observations to the final structure [7].

4.3 Abstraction

As CT and cryo-EM are highly related, the aim in this section is to define an abstraction model. The big difference is, that CT is in 2D and cryo-EM in 3D. Mathematically, an extension from 2D to 3D should be theoretically feasible. It is considered a numerical question due to more need of resources as computation gets more expensive. Therefore, an abstract form will be defined. A similar notation than previously is used, with original object $x \in L^2(\Omega)$. Further, original object dimension space is parametrized with D, consequently $\Omega \subset \mathbb{R}^D$. Additionally, dimension of observation space is defined as $D-1$, such that $\tilde{\Omega} \subset \mathbb{R}^{D-1}$.

$$\begin{aligned} y_i &= p_i + \eta_i(\Delta) \quad &, \text{ with } 1 \leq i \leq N \\ y_i &= \left(A(x, \theta_i) + \eta_i\right)(\Delta) \quad &, \text{ with } 1 \leq i \leq N \end{aligned} \tag{4.5}$$

with

- $x \in L^2(\Omega)$: original object
- $A : L^2(\Omega) \to \mathbb{R}^M), x \mapsto A(x; \theta_i)$: a non-linear operator
- $\theta_i \in SO(P)$: projection angle(s) vector, with P projection dimension
- $\eta \sim \mathcal{N}(0, \sigma^2 I) \in \tilde{\Omega}^M$: i.i.d Gaussian noise
- $\Delta \subset \tilde{\Omega}^M$: term for discretization

Reconstruction: TODO: equation 3.7, Delta is Omega tilde power M not power D. Also, as before, theta is in SO(3) (in equation 3.3 also theta should be in [0,2pi], maybe with finding a consistent definition)
Further, an abstract form of the reconstruction operator is defined as:

$$Recon : L^2(\tilde{\Omega}) \to L^2(\Omega), y \mapsto Recon(y; \theta) \tag{4.6}$$

with

- $\theta_i \in \mathbb{R}^P$: projection angle(s) vector, with P projection dimension

Classical tomography: Classical tomography parameters are defined with $D = 2$, $P = 1$. Further, $A(\cdot)$ is the Radon Transform (see Equation 4.1). Reconstruction operator can be defined as FBP (with or without U-Net).

[7] For further details [1, 8]

Cryo-EM: Cryo-EM parameters are defined with $D = 3$ and $P = 3$ as θ_i not only corresponds to a projection angle vector but also some rotation. Further, $A(\cdot)$ can be defined as Π_z ($Rot(\ x; \theta)$) where Rot is the 3D rotation and Π_z the tomographic projection.

High noise regime: Cryo-EM observations are highly noisy, which makes reconstruction challenging. There are different ways to reduce noise from observations, most of them are related to averaging. Averaging needs to consider similar observations and ignore diverse ones. In the defined abstract model, averaging over paired observations from θ should be a good averaging model.
One idea would be to measure distances between observations. Another way is to find a low-dimensional embedding which maps observation y to θ. When talking from low-dimensional embeddings, there is no way around Graph Laplacian and Graph Learning, which will be introduced in the next chapter.

5
Manifolds and Graphs

In this chapter, the connection between graphs and Graph Laplacian (GL) embedding to CT and cryo-EM is examined. First, some graph foundations are presented. Then, GL and its embedding is introduced and the connection to CT and cryo-EM is illustrated. Further, the term Graph Denoising is defined and last, some Graph Deep Learning approaches are presented.

5.1 Graph Foundations

Real world data is often in a graph structure, like social networks, citation networks, protein interaction networks or a simple google search. If data is not available as a graph structure, a graph can be artificially constructed with k-NN or other methods.

Graph Learning: Graph Learning got a lot of attention in recent years. The idea is to learn graph information, such as topology or connections between nodes, to solve tasks. Attention mechanisms are popular at the moment[21] and the idea was derived to graphs as well [22]. It resumes to computing node features by using local information, therefore, the neighborhood of nodes is exploited. Popular learning tasks are *node classification* or *link prediction*, where a model is learned from node and edge features as well as topology. The model can be used for prediction or classification on nodes or edges. Another common task is *community detection*, which aims to identify clusters of nodes within the input graph. Further, graphs are highly favored for *dimensionality-reduction*, where graph algorithms provide a helpful tool, as ordinary algorithms like principal component analysis fail to establish a meaningful dimensionality reduction.

Constructing Graphs for molecular-imaging: For a cryo-EM or CT observation, a graph can be constructed as well. Every observation y_i can be assigned to a node v_i, consequently $v_i \in \mathbb{R}^M$ and $|V| = N$.
To determine distance between two nodes, a distance measure needs to be defined. For CT, it can be set up by using the $\ell 2$-norm $\|y_i - y_j\|$. A cryo-EM distance measure is more challenging to setup, as projections are drawn with some random 3D rotation and projection.

It can happen that two observations are equivalent up to a 2D rotation. Consider a first observation y_1, which has no 3D rotation and a second observation y_2 with a rotation in x-y plane by 45°. The two projections have a defined in-plane rotation g, such that $g\ y_1 = y_2$. Therefore, a term of in-plane rotation is added to the $\ell 2$-norm: $min_{g \in SO(1)} \|g\ y_i - y_j\|$, which is inspired by [9].

5.2 Graph Laplacian & Manifolds

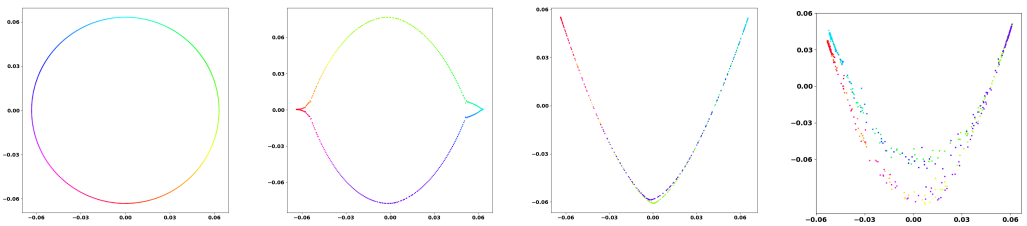
In the following section, the connection of GL and manifolds to CT and cryo-EM is established.

5.2.1 Graph Laplacian

GL is a matrix that represents a graph and can be used to find many important properties. It is a very powerful tool and a good introduction can be found in [19, 23].
GL is defined as follows: $L = D - A$, where A is the adjacency matrix and D the degree matrix (diagonal matrix with degree of nodes as entries).

5.2.2 Graph Laplacian Embedding for CT and cryo-EM

A basic introduction to GL embedding and how it can be computed is found in B.1. In this section, the connection to CT and cryo-EM is established.
For a given observation, a low-dimensional embedding can be computed by using the GL. In the following, it will be observed how this embedding for CT looks like. To be more precise, the second and third smallest eigenvectors of GL will be observed. Shepp-Logan phantom is used again as an example for CT. For Radon Transform, θ and s are specified with $\theta \in \mathbb{R}^{500}$ as evenly spaced between $[0, 2\pi]$ and $dim(s) = 200$.
In Figure 5.1(a), embedding calculated from clean sinogram and $k = 2$ can be seen. It looks like a perfect circle and angles are in order. Further, noise was added to the sinogram to reach SNR_y 20 dB, 10 dB and 0 dB. Additionally, the embedding was computed with $k = 6$ and illustrated in Figure **??**, Figure **??** and Figure **??** respectively. For SNR_y 20 dB, a circle like object could be established and for SNR_y 10 dB it looks like a half circle.
TODO: adjust ticks for 0dB

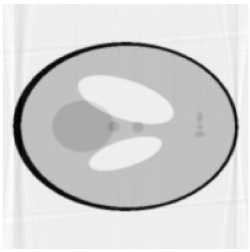


(a) Clean sinogram embedding with $k = 2$ (b) GL embedding $k = 6$ with SNR_y 20 dB (c) GL embedding $k = 6$ with SNR_y 10 dB (d) GL embedding $k = 6$ with SNR_y 0 dB

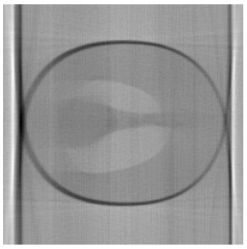
Figure 5.1: Shepp-Logan phantom sinogram GL eigenvectors

In the fields of CT and cryo-EM, underlying low-dimensional embedding from GL is well-defined for noiseless data. In 2D the underlying GL-manifold is a circle, whereas in 3D the GL-manifold is defined as a sphere. This fact can be exploited during learning.

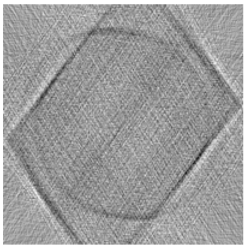
Tomography for unknown angles: But what can we use this embedding for? It is defining a low-dimensional mapping from high-dimensional space. In the case of CT and cryo-EM, this embedding approximates angles for observations. Therefore, for (noisy) observations, angles can be approximated and reconstruction can be established even if angles are unknown. The problem here is the quality of our embedding. As long as the computed embedding is a mapping to the circle (or sphere), it should be reasonable to do reconstruction with. In Figure 5.2 reconstruction with unknown angles is applied. Again, $k = 6$ but SNR_y 20 dB and 0 dB are used. Clean reconstruction with approximated angles in Figure 5.2(a) looks good, despite that there is an in-plane rotation of around 45 degrees. But even more moderate noise 20 dB, where the embedding almost looks like a circle (Figure 5.1(b)) the difference between reconstruction with known angles (Figure 5.2(d)) and unknown angles (Figure 5.2(b)) is pretty big. For 0 dB, already with known angles (Figure 5.2(e)), reconstruction fails and with unknown angles barely anything from original Shepp-Logan phantom can be determined (Figure 5.2(c)).



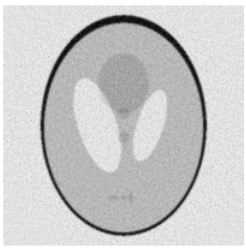
(a) Clean sinogram reconstruction from GL estimated angles.



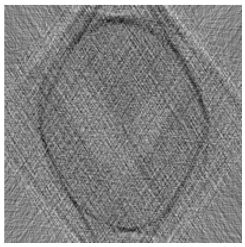
(b) Noisy sinogram with SNR_y 20 dB reconstruction from GL estimated angles.



(c) Noisy sinogram with SNR_y 0 dB reconstruction from GL estimated angles.



(d) Noisy sinogram with SNR_y 20 dB reconstruction from known angles.



(e) Noisy sinogram with SNR_y 0 dB reconstruction from known angles.

Figure 5.2: Shepp-Logan phantom sinogram GL eigenvectors

Not only noise is reducing GL embedding quality, also k, number of observations and observation dimension have an impact to the final result. Some further considerations are presented in Section B.1.1.

With the GL embedding of observations, angles can be approximated. Therefore, CT and cryo-EM problem can be solved for unknown angles, even tough the quality is slightly worse.

Denoise observations: As the quality of reconstruction is highly dependent on observations, it is expected to increase when the level of noise is decreased. Therefore, standard denoising methods like Block-matching and 3D filtering (BM3D) [7] or non-local means [3] could be used to denoise observations to get higher quality reconstructions. Both algorithms emerged from Signal Processing and are not operating on a graph structure. But, they use a neighborhood for averaging, which shows great potential for graph as a data structure for denoising, as graphs can represent neighborhoods really well. BM3D is considered the state-of-the-art denoising algorithm, before algorithms emerged from Deep Learning. Therefore, it will be used as a baseline algorithm in the practical part. Further, as illustrated with the GL embedding, graphs can restore information with a suitable prior, such as angles are uniformly sampled.
In this Thesis,

5.3 Graph Denoising

Graph Denoising is not a common term in literature. In the current chapter, a way of constructing a k-NN graph from observations was introduced. Moreover, the underlying true graph was found by computing the GL embedding, which is a circle or a sphere. Therefore, our constructed graph from observations can be considered a noisy graph, as observations are noisy. As a consequence, the constructed noisy graph has some missing and additional edges compared to the true graph. The goal of Graph Denoising is to estimate the original graph G from a given noisy graph G_0. In other words, noisy graph G_0 will be denoised. This is my definition for Graph Denoising, which is rather related to signal or image denoising. Reconstruction of a true signal given noisy observation signal is done via averaging, that can be performed
For every noisy graph there exists an original graph $G = \langle V, E\rangle$. The noisy graph G_0 can further be defined as $G_0 = \langle V, E_0\rangle$, where $E_0 = E \setminus E_0^- \cup E_0^+$ with $E_0^- \subseteq E$ and $E_0^+ \cap E = \emptyset$. G_0 consists of same nodes V as original graph G. From E some edges are removed (denoted by E_0^-) and some are added (denoted by E_0^+), which results in edges E_0.

Connection to link prediction: Link prediction is a common task in Graph Learning. The goal is to predict existence of a link between two nodes. The task can be formulated as a missing value estimation task. A model M_p is learned from a given set of observed edges. The model finally maps links to probabilities $M_p : E' \to [0, 1]$ where E' is the set of potential links.

Further, U determines the set of all possible edges of G, therefore $E \subseteq U$. Clearly, Graph Denoising can be seen as a link prediction problem. The difference is, that in link prediction a model from a set of observed links is learned $E' \subseteq E$ and in Graph Denoising model is learned from $E' \subseteq U$. Link prediction problems are a subset of graph denoising problems.

5.4 Graph Deep Learning

Graph Denoising can be seen as a way of link predication. The state-of-the-art methods for solving link prediction are *Graph Deep Learning* approaches. With Graph Neural Networks (GNN) [11] the framework for neural networks with graphs has been established.
Using Graph Convolutional Networks (GCN) [13] for graph feature extraction is a popular way. With GCN a new feature representation is iteratively learned for node features (edge features are not considered). It can be seen as an averaging of nodes over their neighborhood where all neighbors get the same weight combined with some non-linear activation (e.g. ReLu). To consider the node itself in averaging, Kipf and Welling [13] applies the so-called "Renormalization trick", where self-loops are added to the adjacency matrix and after every layer, a normalization step is applied. The topology of the graph will not be adjusted during the learning process.
Simple Graph Convolutional Network [25] proposed a simplified version of GCN. They could verify their hypothesis that GCN is dominated by local averaging step and non-linear activation function between layers do not contribute too much to the success of GCN. Therefore, it can be seen as a way of power iteration[8] over the adjacency matrix with normalization in every layer. Wang et al. [24] proposed an extension to GCN by not operating on the same graph in every layer but adopting underlying graph topology layer by layer. Graph Attention Networks (GAT) [22] extended the concept of GCN with attention where not all neighboring nodes get the same weight (attention). Again, topology of the graph will not change but weighted averaging over the neighborhood will be computed and this is what in denoising is a good idea.

[8] For more details see A.2

6
GAT-Denoiser

In this Chapter, I introduce my methodological approach. As a result, a GNN is derived which is called *GAT-Denoiser*. Its main components and overall architecture is introduced. GAT-Denoiser was implemented for 2D CT and therefore, some notes about CT are given.

Goal: CT and cryo-EM in the high-noise regime is the domain of interest. For a given set of observations, a denoising model is sought, such that it enables denoising of observations such that reconstruction quality will increase. As a first step towards an algorithm which works for unknown observation angles, angles are fixed during practical part of this Thesis.

Input graph: As θ is fixed, angle corresponding to each observation are known. Therefore, a graph can be constructed to model neighboring observations based on their angles. To define a distance measure, angles can be mapped to the unit circle (or sphere). Then, a geodesic distance can be computed with the great-circle distance. Based from these distances, a k-NN graph can be constructed.

For GAT-Denoiser, this entails that graph topology is fixed and a k-NN graph can be constructed from θ.

6.1 Pipeline

In the following section, the GAT-Denoiser pipeline is introduced. The pipeline consists of three neural network parts, namely convolution, GAT and U-Net. For readers who are not familiar with these concepts, Chapter C present an introduction.
GAT-Denoiser is a GNN and has two main components, namely convolution layers and GAT layers. The main idea of GAT-Denoiser is to enable denoising of observations:

$$\textit{GAT-Denoiser}(\cdot) : L^2(\tilde{\Omega}) \to L^2(\tilde{\Omega}), y \mapsto \textit{GAT-Denoiser}(y) \tag{6.1}$$

Input y of GAT-Denoiser is a noisy observation and output is a denoised version. GAT averages over observation neighbors and convolution denoise single observations. For every

GAT layer there is a preceding convolution. In the case of CT, convolution is in 1D, where for cryo-EM it is in 2D.

The GAT denoise observation signal with its neighbors by averaging. Further, convolution is added to denoise single observations.

But, the main overall goal is to get the best possible reconstruction from noisy observation y which approximates original object x and not just denoise observation y which approximates noiseless observation p.

$$x \approx Recon\left(GAT\text{-}Denoiser\left(y\right)\right), \tag{6.2}$$

Therefore, an end-to-end learning approach is used where quality of reconstruction is compared during GAT-Denoiser training, which is expected to perform better than only optimizing denoising of observations.

TODO: remove header, remove FBP + U-Net In Figure 6.1 overall GAT-Denoiser pipeline is illustrated.

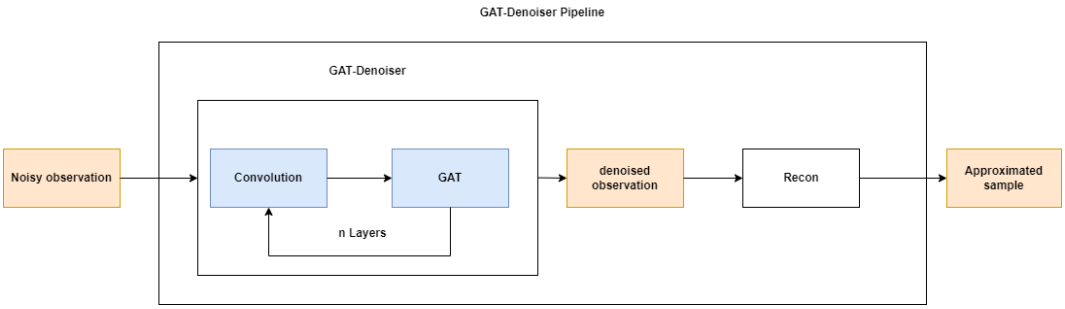


Figure 6.1: GAT-Denoiser pipeline

6.2 Layers

In the following, a closer look at the neural network layers will be given.

Input of the layers are observations and output are denoised observations. Therefore, input and output dimension is defined as $\mathbb{R}^{N\times M}$.

Figure 6.2 presents the detailed GNN architecture. It is parametrized with *channels*, *heads* and *layers*. The number of channels in convolution can be increased with parameter *channels*. Further, *heads* determines the number of heads used in the GAT layers and parameter *layers* defines how many convolution and GAT layers are stacked together.

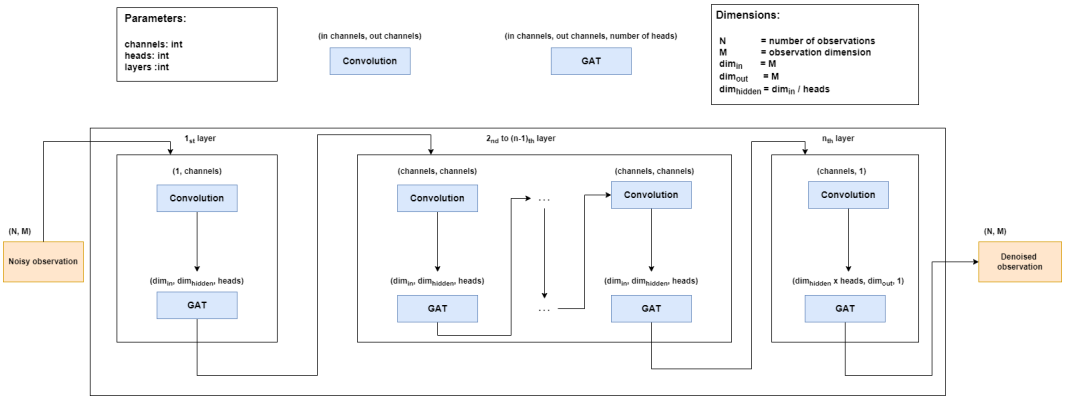


Figure 6.2: Overall GAT-Denoiser architecture

For every layer, first convolution and then GAT is processed. Convolution in the whole network was defined with kernel size of 3 and padding 1, therefore, size of convolved signal will not change. Convolution can be defined with different parameters, but output signal needs to have same size as input signal. Further, additional convolutional channels can be used for learning. If parameter $channels > 1$, channels are increased in the first convolution layer and decreased in the last one. Parameter *heads* controls multi-head approach for GAT. Input and hidden dimension of GAT is M if no heads are used. If multi-head attention is used, hidden dimension will be set to $M/heads$. In the last GAT layer, everything gets prepared for output dimension and averaging with 1 head is applied.

K-hop neighborhood: In GNNs, multiple layers expose the k-hop neighborhood. So for a network with k layers, network operates on the k-hop neighborhood. In GAT-Denoiser, this corresponds to the layers of GAT. Therefore, if writing from one layer, it is referring to convolution and GAT together.

6.3 Training

For GAT-Denoiser, one could think of two different losses to use during training. First, one could think of defining the loss on an observation level directly, what I denote by $\mathcal{L}_{sino}$. The $\ell 2$-norm is giving a meaningful measure:

$$\mathcal{L}_{sino} = \| \, p_i - \mathit{GAT\text{-}Denoiser}(A(x_i, \theta, s) + \eta) \, \|_2^2 \tag{6.3}$$

Second idea is to use an end-to-end learning approach where quality of reconstruction is compared in the loss. Thus, the output of GAT-Denoiser is not directly part of the loss, but first reconstruction will be computed. As the quality of reconstruction is measured in the loss, resulting model is expected to be optimized for reconstruction quality, and not observation quality. But, it is expected to be more computing expensive, as reconstruction is needed to calculate loss during training.

$$\mathcal{L} = \| \, x_i - \mathit{Recon}(\mathit{GAT\text{-}Denoiser}(A(x_i, \theta, s) + \eta)) \, \|_2^2 \tag{6.4}$$

As x_i is part of loss $\mathcal{L}_{sino}$ and $\mathcal{L}$, access to original object is needed during training.

Dropout: Dropout is used in neural network as regularization and to prevent over fitting. During training the neural network, some units are randomly omitted. Therefore, the network is considered to not over fit to single unit, as during learning they are not always present. There is a dropout parameter in GAT, which can be considered during training.

Reconstruction Computed Tomography: Reconstruction in the CT case is defined with U-Net, as $Recon : UNet(FBP(\cdot))$. Thus, U-Net needs to be first pre-trained with the desired dataset. Then, in a second step, one could consider to jointly train U-Net with GAT-Denoiser.

7 Results

In this chapter, results of GAT-Denoiser are presented. First, the LoDoPaB-CT dataset is introduced, and some illustrations are given. Second, project setup and shared settings are given. Third, small scale GAT-Denoiser experiments are presented, where the goal is to find good parameters for training the GAT-Denoiser model. Last, the large scale GAT-Denoiser experiments are presented, where the goal is to find the best model.

7.1 Dataset

GAT-Denoiser is tested on the LoDoPaB-CT [14] dataset, which is a benchmark dataset for low-dose CT reconstruction methods and therefore well suited for our domain.
The dataset consists of 35'820 train images and 3'553 test images. All these images are having resolution 64x64.

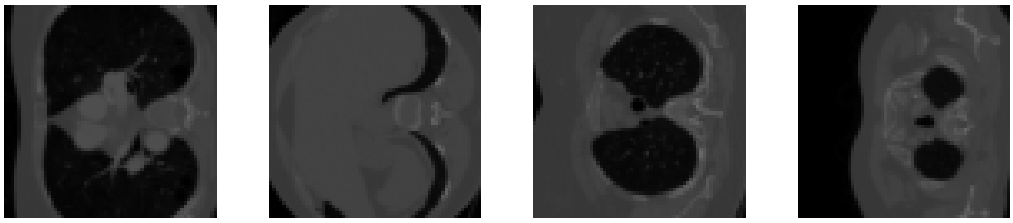


Figure 7.1: Some training images of LoDoPaB-CT dataset.

7.2 Project setup

Python source code of the project is available on GitHub[9]. Several python packages are used in the code and the most important one are listed: Pytorch geometric[10] for neural network configuration, Operator Discretization Library[11] for Radon Transform and FBP, and last, Weight & Bias[12] (wandb) to gather results in a convenient way.

[9] https://github.com/cedricmendelin/master-thesis
[10] https://pytorch-geometric.readthedocs.io/en/latest/
[11] https://odlgroup.github.io/odl/
[12] https://wandb.ai/site

Training of GAT-Denoiser has been performed on the HPC-cluster scicore of the University of Basel. During training, up to 4 titanx GPUs with each 12 GB RAM have been used.

Radon Transform: For Radon Transform, sampling points have been fixed to 64, thus $s \in \mathbb{R}^{64}$. Further, projection angles θ are sampled within interval $[0, \pi]$. The number of observation is the size of our input graph nodes, where typically 1024 are used.

GAT settings: During experiments, in the GAT layers exponential linear unit (ELU) was used as an activation function. Further, dropout is used as regularization term. During training, Adam [12] optimizer was applied with learning rate 0.01 and weight decay 0.0005. Moreover, mini batch gradient descent with a batch size of 64 was computed.

Performance metrics: During evaluation, two main metrics are considered. First, *Loss* is used, which refers to the average ℓ-2 distance between original object and reconstruction. Second, *SNR* is used, which refers to SNR in dB of reconstruction, compared to original object. Further, visual results are presented and can be seen as a third metric.

U-Net training: U-Net was pre-trained on the LoDoPaB-CT training dataset, with 128 channels in the first contracting step. In total, 4 steps have been computed, which results in 1024 channels as output of last contracting step. During training, noise was sampled from normal distribution to reach SNR in the interval $[0, -10]$ dB and was added to the observation. In total, the model was trained for 200 epochs.

BM3D: In the GAT-Denoiser pipeline, first, observations will be denoised, and second, reconstruction is computed. Therefore, BM3D can be applied at two different steps. First, it can be used to denoise sinogram and forward denoised sinogram to FBP, which is how GAT-Denoiser work. Secondly, FBP can be computed with noisy sinogram and output can be forwarded to BM3D, which will denoise reconstruction. In the following term *BM3D sino* and *BM3D reco* are used to distinguish between the two approaches.

7.3 Small scale GAT-Denoiser experiments

For small scale experiments, not the complete LoDoPaB-CT dataset was considered, but only 1024 train images and 100 validation images. Further, evaluated GAT-Denoiser models presented in this section have been trained for 200 epochs.

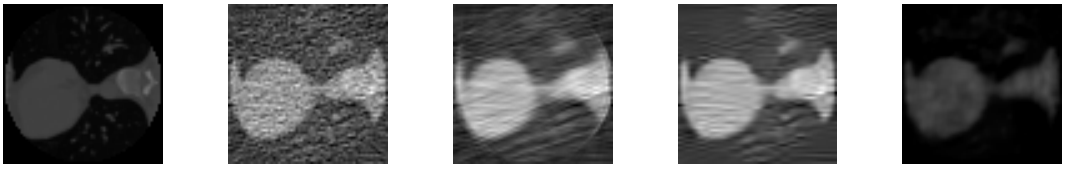
7.3.1 Baseline

Table 7.1 shows baseline results for FBP, BM3D sino, BM3D reco and U-Net. Noise was added to reach SNR_y of 0, -5, -10 and -15 dB.
As expected, reconstruction with FBP performs the worst, as it only reconstructs from noisy observation. BM3D-sino and BM3D-reco perform surprisingly good for higher SNR_y . Although, for lower SNR_y the algorithm performs poorly as well.

Figure 7.2 illustrates the reconstruction of a single observation for all baseline algorithms. SNR_y is set to 0 dB.
Visually U-Net reconstruction looks best, but the SNR was lower compared to the BM3D approaches. After some investigation, a small bug in the pre-trained U-Net model was found. There was an issue with the amplitude of U-Net reconstruction, therefore the SNR values of U-Net are expected to be slightly higher. But also the pre-trained U-Net model starts to fail reconstruction with lower SNR_y .

Algorithm	**SNR_y 0 dB**		**SNR_y -5 dB**		**SNR_y -10 dB**		**SNR_y -15 dB**	
	SNR	**Loss**	**SNR**	**Loss**	**SNR**	**Loss**	**SNR**	**Loss**
FBP	4.50	1182.3	-0.31	1982.2	-5.25	3454.1	-10.22	6101.7
BM3D-sino	9.93	714.2	7.42	892.2	4.61	1179.8	2.00	1570.1
BM3D-reco	10.79	664.5	8.09	833.6	4.97	1137.7	1.54	1677.5
U-Net	7.13	977.7	6.18	1054.1	4.34	1235.0	1.87	1545.4

Table 7.1: Baseline results small experiments: the best result in column is marked bold.



(a) Clean image (b) FBP reconstruction (c) BM3D sino reconstruction (d) BM3D reco reconstruction (e) U-Net reconstruction

Figure 7.2: Sample reconstruction for baseline algorithms and SNR_y 0 dB

7.3.2 K-NN Experiments

In the following section, experiments with different input graphs are presented. Therefore, some other parameters are fixed for all experiments. First, U-Net was deactivated, resulting reconstruction is defined as FBP solely. Further, GAT-Denoiser has been set to 3 layers with GAT single head and convolution with one channel, kernel size 3 and padding 1. If nothing else noted, noise was added to reach SNR_y 0 dB.
The input graph structure is important for GAT learning. In our case, GL embedding for the observation is expected to be a circle and therefore, the input graph was fixed. A k-NN graph was built from angles, which are mapped to the unit-circle and great-circle distance was used to computed distances. Moreover, learning is expected to fail when learning on a random graph.
Next to the input graph, the neighborhood size for k-NN graphs is explored, as it is expected to be not easy to find good values for k.

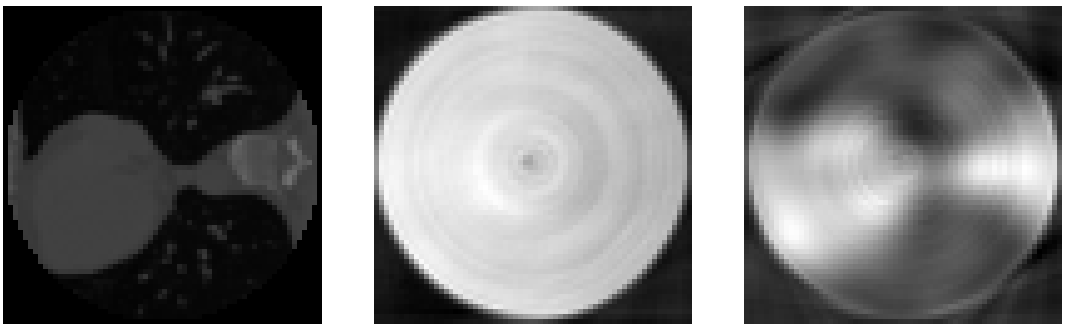
Does learning fail with random graph? Yes it does.
During this experiment, two models with different input graphs, but same number of nodes 1024 have been trained. First, a k-NN graph with $k = 10$ was defined as input, therefore,

around 1% of all nodes are connected to a single node. Second, a random Erdős–Rényi graph with $p = 0.01$ has been evaluated, where every node is connected to every other node with probability p. As a consequence, nodes are approximately connected to 1% of all available nodes as well.

In the experiment, random Erdős–Rényi graph fails to learn denoising and k-NN starts to capture first details. Table 7.2 shows Loss and SNR and in Figure 7.3 example reconstruction is illustrated. The reconstruction quality of GAT-Denoiser in Figure 7.3(c) is rather low. However, it starts learning a few coarse details and with the following experiments, the reconstruction can hopefully be increased in quality. But, our assumption for GL embedding to be on a circle is reasonable.

Input graph	**Loss**	**SNR**
Erdős–Rényi graph with $p = 0.01$	1267.16	3.65
k-NN graph with $k = 10$	804.7	7.3

Table 7.2: Loss and SNR for random graph vs. k-NN graph.



(a) Clean sinogram embedding with $k = 2$

(b) Reconstruction Erdős–Rényi input graph with $p = 0.01$

(c) Reconstruction k-NN input graph with $k = 10$

Figure 7.3: Example image reconstructions for different input graphs.

Does result improve with more observations? Yes. For this question to answer, multiple models have been trained for graphs with 512, 1024 and 2048 observations as well as different $k = (2, 6, 8, 10, 20)$. In Table 7.3 results are presented, where results are averaged by number of observations for different k.

A larger number of nodes can improve the models. But, training is also more expensive regarding execution time. Therefore, for all upcoming experiments number of observations is fixed to 1024, as it looks like a good value in between, with good reconstruction result but also not too much training time.

Number of observations	SNR	Loss	Training time (s)
512	7.22	822.6	2678.6 s
1024	7.80	766.88	4216.1 s
2048	8.55	701.49	7609.0 s

Table 7.3: Loss and SNR for different number of observations, which refer to the average of different k-NN parameters $k = (2, 6, 8, 10, 20)$.

What is a good value for k? To find suitable k, multiple models have been trained with different $k = (1, 2, 3, 4, 6, 8, 10, 20)$ as well as different SNR_y (0 dB and -10 dB).
In Table 7.4 the results of evaluated models are presented. For SNR_y 0 dB the best result can be achieved with $k = 2$ and for SNR_y -10 dB $k = 1$. With our assumption of fixed angles, it looks like only a few neighbors are enough to capture important information in the neighborhood.

k-NN parameter k	SNR_y 0 dB SNR	0 dB Loss	SNR_y -10 dB SNR	-10 dB Loss
1	8.81	678.67	**7.09**	**827.51**
2	**9.16**	**652.38**	6.95	849.44
3	8.64	689.15	6.38	912.73
4	8.37	712.44	6.29	906.75
6	8.25	722.81	5.81	958.32
8	7.99	744.07	5.64	985.81
10	7.30	804.70	5.54	996.25
20	6.30	910.45	4.82	1090.55

Table 7.4: Different k-NN values for SNR_y 0 dB and -10 dB

7.3.3 GAT Experiments

In this section, experiments are presented with focus on the GAT and its parameters. Mainly, two parameters are important: number of heads and number of layers.
Therefore, number of layers is expected to keep rather low, as it defines the exposure of the k-hop neighborhood. The dataset observations have only resolution 64x64 and consequently, it is expected that too many layers will average a too large neighborhood.
Further, number of heads determines if the learned weight matrix is divided into several parts. The larger the number of heads, the smaller are the parts of the weight matrix. Overall, larger number of heads is expected to smoothen denoising, but too large values will divide weight matrix in too many parts, leading to bad results.
Experiments have been computed in parallel and results for k-NN have not been present before. Therefore, graphs were constructed with $k = 8$, even tough it is not the best value for k. Further, convolution was applied with a single channel, kernel size = 3 and padding 1. As focus is on GAT parameters, U-Net was again omitted. Further, multiple experiments with layers = $(2, 3, 4, 6)$ and heads = $(1, 2, 4, 8, 16)$ have been evaluated, where results can be found in Table 7.5.

Heads	2 Layers		3 Layers		4 Layers		6 Layers		Average	
	SNR	**Loss**	**SNR**	**Loss**	**SNR**	**Loss**	**SNR**	**Loss**	**SNR**	**Loss**
1	7.55	784.23	6.65	869.1	**6.08**	**934.2**	3.19	1334.1	5.87	980.4
2	7.88	753.4	7.17	821.7	5.98	958.8	3.55	1256.5	6.15	947.6
4	8.22	727.55	6.26	927.4	4.10	1176.1	**4.08**	**1191.2**	5.67	1005.6
8	8.14	732.28	**7.28**	**808.1**	5.54	1010.4	2.81	1516.2	5.94	1016.7
16	**8.27**	**721.9**	5.69	990.5	5.98	941.8	2.81	1516.0	5.66	1042.6
Average	8.01	743.9	6.59	883.4	5.54	1004.3	3.29	1362.8		

Table 7.5: Models trained with different GAT parameters: best and average result in column is marked bold.

The more layers the better result? No. As expected, when working with too many layers, denoising starts failing as it averages over a neighborhood, which is too large. In Table 7.5, 6 layers clearly get the worst result, followed by 4, 3 and 2 layers.

How does multiple heads affect result? It has an overall positive impact. During experiments, for every layer l with single-head result, there is a better result for l, but multiple-heads. For 2 layers, single-head model performs with an average SNR of 7.55 dB where with 16 heads average SNR is 8.27.
2 layers seems to be most promising and number of heads 4 seems to be a reasonable choice for heads. With 16 heads, the result was even a bit better. As 16 heads performed rather low for other number of layers, the best GAT parameters are considered to be 2 layers and 4 heads. To make sure that $k = 2$ is a good setting for these GAT parameters as well, the k-NN experiment have been recomputed for 2 layers and 4 heads. Table D.2 presents results, which give the same insight as for previous k-NN experiments.

How about some dropout? A small experiment with fixed network parameters and different dropout values $(0, 0.01, 0.03, 0.05, 0.1)$ have been calculated. Results are presented in Figure D.3. There is not too much of a difference, without dropout, model got the best value, followed by 0.03 and 0.05 with almost the same result.
Therefore, in future trainings, dropout of 0.05 was used.

7.3.4 Convolution Experiments

For convolution, kernel size and padding can be defined. Moreover, it is possible to increase number of channels during convolution. For GAT, number of layers determine the averaging neighborhood. As multiple channels are expected to perform better with multiple layers, there could be a trade-off between GAT quality (size of averaging neighborhood) and convolution quality (increasing number of channels).
First, some experiments with 4 layers and 2 heads have been executed with different number of kernel, padding and number of channels. Noise was added to reach an SNR of -5 dB. The aim is to find good convolution kernel and padding parameter value.

What is a good kernel size? In Table 7.6 results are illustrated. Best results are achieved with only 1 channel and smallest kernel size of 3 with padding 1. This is not too big of a surprise, observation dimension is only 64 and, therefore, a smaller kernel make sense.

Kernel and Padding	1 Channel		4 Channel		8 Channel		16 Channel	
	SNR	**Loss**	**SNR**	**Loss**	**SNR**	**Loss**	**SNR**	**Loss**
7 / 3	3.56	1328.7	4.60	1330.0	**3.76**	**1363.4**	**2.81**	**1515.7**
5 / 3	4.94	1081.0	2.76	1548.5	2.66	1597.0	**2.81**	**1515.7**
3 / 1	**6.49**	**900.6**	**5.06**	**1084.8**	1.30	2336.1	0.05	2844.4

Table 7.6: Small experiment: Convolution for GAT-Denoiser with 4 layers and 2 heads

Is there a trade-off between GAT and convolution quality? As kernel size of 3 with padding 1, seems to be a good value for the LoDoPaB-CT dataset, in the second convolution experiment different GAT architectures are compared with kernel size 3. Good performing GAT architectures for layers 2 to 4 have been chosen: 2 layers with 4 heads, 3 layers with 8 heads and 4 layers with 2 heads.
Resulting evaluation can be seen in Table 7.7. Additionally, for every GAT architecture model without convolution was computed as well. For every architecture, there was a model with convolution which performed better than the one without convolution.

Convolution can be considered to contribute to the success of GAT-Denoiser.

Channels	2 Layers		3 Layers		4 Layers	
	SNR	**Loss**	**SNR**	**Loss**	**SNR**	**Loss**
No Conv	6.99	839.3	4.31	1175.5	**6.91**	**842.2**
1	**8.10**	**735.6**	6.91	**842.2**	6.49	900.6
4	7.12	834.6	6.45	903.0	5.06	1084.8
8	7.79	767.3	**7.01**	862.9	1.30	2336.1
16	7.45	798.8	5.97	1002.9	0.05	2844.4

Table 7.7: Small experiment: Convolution for GAT-Denoiser with different number of layers and heads

7.3.5 Loss Experiments

An experiment was set up to compare $\mathcal{L}$ from Equation 6.4 with $\mathcal{L}_{sino}$ from Equation 6.3. The same network parameters have been used, but once loss during training was defined with $\mathcal{L}$ and once $\mathcal{L}_{sino}$. The experiment have been computed for SNR_y 0, -5, -10 and -15 dB.

Is our end-to-end learning a good idea? Yes, but it comes with a price. Averaged results over training loss can be found in Table 7.8. Overall, model trained with loss $\mathcal{L}$ performed around 1 dB in SNR better. But, training of the network took much longer,

more than factor 3. This is not a big surprise, as for learning with $\mathcal{L}$, reconstruction of every sample needs to be done in every epoch, where for $\mathcal{L}_{sino}$ reconstruction can be omitted.

End-to-end learning with loss $\mathcal{L}$ contributes to the success of GAT-Denoiser and is used in further experiments.

Loss training	Average SNR	Average Loss	Training time (s)
$\mathcal{L}$	7.29	816.7	3812 s
$\mathcal{L}_{sino}$	6.49	981.3	1209 s

Table 7.8: Average Loss and SNR of different training loss $\mathcal{L}$ and $\mathcal{L}_{sino}$

7.3.6 GAT-Denoiser Component Experiments

The final small scale experiments aim to see contribution of the single components in action and to compare them. Therefore, a fixed overall architecture is defined and experiments with different combination of components (Convolution, GAT and U-Net) will be calculated. Further, models with joint U-Net training have been evaluated. The fixed settings for GAT-Denoiser have been defined with 2 layers, 4 heads, convolution with kernel size 3 and padding 1. k was set to 2. Overall, 8 models have been evaluated and compared against the baseline algorithms for $\text{SNR}_y = (-15, -10, -5, 0)$ dB .

Results are plotted in Figure **??** and numeric values can be found in Table D.1. First, *GAT* is the model, where no convolution and no U-Net is active. Second, *GAT + Conv* refers to the model with GAT and convolution but no U-Net. Additionally, the two models have been combined with U-Net, which are referred as *GAT + U-Net* and *Conv + GAT + U-Net*. Last, models with joint U-Net training are presented. *U-Net** denotes that during all 200 epochs, U-Net was jointly trained. *U-Net*** denotes that during the first 100 epochs, U-Net was kept static and training was activated for the second 100 epochs.

TODO: fix plot, same solor for some values D.1

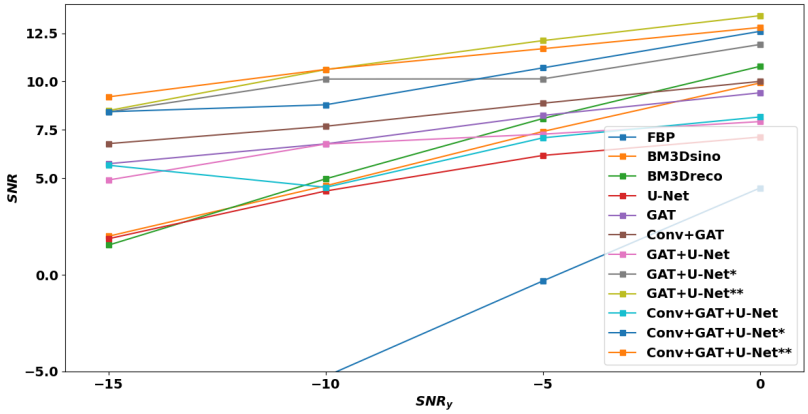
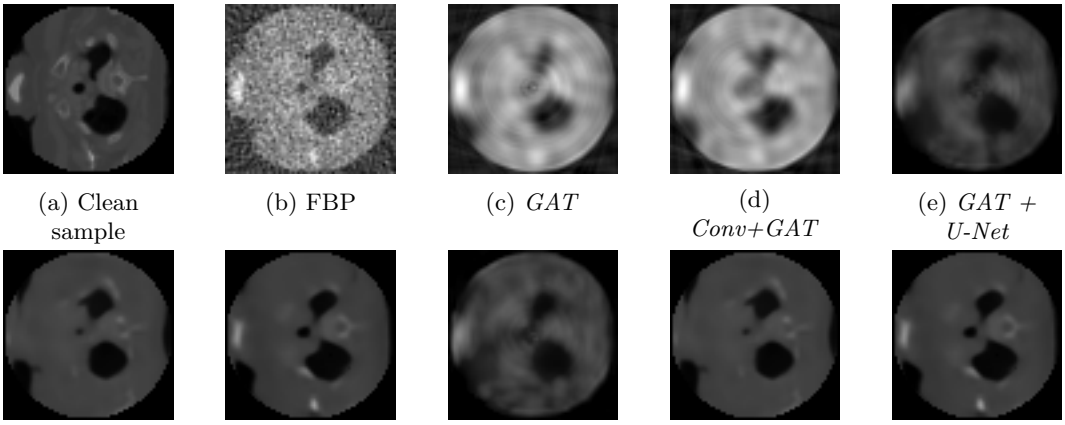


Figure 7.4: Reconstruction SNR of different models and baseline algorithms.

Model *GAT* is not able to beat baseline algorithms, and it seems that only applying GAT is not enough. *GAT + Conv* performs slightly better. When adding U-Net in the models *GAT + U-Net Conv + GAT + U-Net*, the visual results improve slightly but reconstruction SNR is still not able to beat baseline algorithms. Finally, with joint training of GAT-Denoiser and U-Net, results can beat baseline algorithms for the small scale problem. It was helpful, to first train only GAT-Denoiser for 100 epochs and start joint training for the second 100 epochs.
Figure 7.5 illustrates visual results for all baseline algorithms and some GAT-Denoiser models. Further, clean sample is presented in Figure 7.5(a).



(a) Clean sample | (b) FBP | (c) *GAT* | (d) *Conv+GAT* | (e) *GAT + U-Net*

(f) *GAT + U-Net** | (g) *GAT + U-Net*** | (h) *Conv + GAT + U-Net* | (i) *Conv + GAT + U-Net** | (j) *Conv + GAT + U-Net***

Figure 7.5: Sample reconstruction for small scale GAT-Denoiser models SNR_y 0 dB

7.4 Large scale Experiments

In this section, large scale experiments are presented, which have been trained on the complete LoDoPaB-CT dataset as well as evaluated on the complete LoDoPaB-CT test dataset. From last section, the most promising parameters for the GAT-Denoiser architecture have been chosen to find the best model. The following parameters have been fixed during the large scale experiments: $k = 2$, number of layers = 2, number of heads = 4, convolution with kernel size 3 and padding 1. The GAT-Denoiser have been trained for 20 epochs, if not mentioned differently.

7.4.1 Baseline

In Table D.4, baseline results for large experiment can be found. As only the number of samples changed, but algorithms are fixed, the result is roughly the same as for the small scale experiments.

7.4.2 GAT-Denoiser components

The main components of the GAT-Denoiser pipeline have been tested with different models. As a result of the small scale experiments, jointly training U-Net was most promising, when there was first some training on GAT-Denoiser solely and in a second part a joint training phase. Therefore, model Conv + GAT + U-Net* and GAT + U-Net* is omitted for the large scale experiments. Moreover, ... U-Net** denotes that models have been first trained 10 epochs without U-Net training and then 10 epochs with U-Net training.

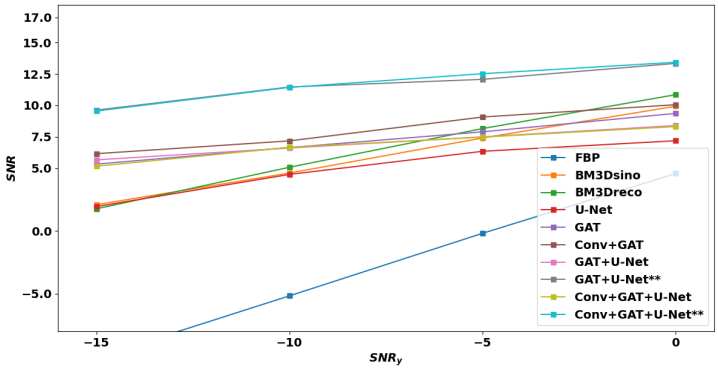


Figure 7.6: Resulting SNR of different models and baseline algorithms.

Figure **??** presents resulting SNR for baseline algorithms and different GAT-Denoiser models. Further, numerical results are presented in Table D.5 The results are rather similar compared to the small scale experiments. Overall reconstruction is slightly better, as during training more samples have been seen. Again, approaches with joint U-Net and GAT-Denoiser training results best.

7.4.3 Best Models

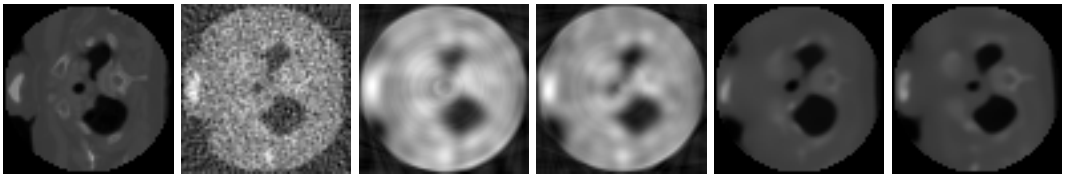
For the most promising architectures, the one with joint U-Net and GAT-Denoiser training, the models have been further trained for another 20 epochs. In Table 7.9 final results are illustrated. The number in brackets denotes the number of training epochs without U-Net followed by the number of epochs with joint model training.

All in all, the few more epochs could sometimes improve result, as for *Conv + GAT + U-Net* and SNR$_y$ 0 dB. Although, this is not always the case and in some scenarios, the model even performed worse, as for *GAT + U-Net* and SNR$_y$ -10 dB.

Model	SNR_y 0 dB SNR	SNR_y 0 dB Loss	SNR_y -5 dB SNR	SNR_y -5 dB Loss	SNR_y -10 dB SNR	SNR_y -10 dB Loss	SNR_y -15 dB SNR	SNR_y -15 dB Loss
GAT+U-Net(10/10)	13.34	413.8	12.07	469.2	**11.46**	**502.0**	9.62	633.5
GAT+U-Net(10/30)	**13.43**	**406.2**	**12.85**	**428.2**	10.83	546.2	9.65	**603.34**
GAT+U-Net(20/20)	13.16	412.2	12.10	469.9	11.02	529.0	**10.03**	606.0
Conv+GAT+U-Net(10/10)	13.43	403.5	12.52	450.1	**11.43**	509.9	9.55	638.0
Conv+GAT+U-Net(10/30)	13.68	395.4	12.32	460.3	11.40	**509.1**	9.25	688.1
Conv+GAT+U-Net(20/20)	**13.87**	**385.4**	**12.81**	**442.3**	10.90	547.5	**10.1**	**604.4**

Table 7.9: Baseline results large experiments

A single sample reconstruction for SNR_y 0 dB is presented in Figure 7.7. Additional visual results are presented in the Appendix D.6 for baseline algorithms and the best performing large scale GAT-Denoiser models for different SNR_y .



(a) Clean sample (b) FBP (c) *GAT* (d) *Conv+GAT* (e) *GAT+U-Net(10/30)* (f) *Conv+GAT+U-Net(20/20)*

Figure 7.7: Large scale experiments visual results.

8

Conclusion and Future Work

In this final chapter, conclusion and future work is presented.

8.1 Conclusion

In this Thesis, the molecular imaging methods CT and cryo-EM within the high noise domain, have been the problems to approach.
During the practical part, 2D and 3D tomography have been implemented in python to familiarize with the domain. Further, vector diffusion map was implemented in 3D to so some comparison and code was integrated with Aspire[13]. Finally, GAT was implemented and further optimized with convolution and U-Net. Therefore, GAT-Denoiser was introduced.
During evaluation on the LoDoPab-CT dataset, baseline algorithm BM3D was implemented. Overall, GAT-Denoiser shows great results and improves reconstruction SNR by 379.9%, 126.0% 57.7%, 27.6% for SNR_y -15 dB, -10 dB, -5 dB and 0 dB. GAT is able to capture important observation information even in the high-noise domain. Moreover, the individual components of GAT-Denoiser contribute to the success of GAT-Denoiser. It was shown that convolution, GAT and the end-to-end learning approach are all valuable individually and are therefore well-chosen. Further, GAT-Denoiser combined with U-Net and joint neural network training can upgrade CT reconstruction to a new level.

8.2 Future Work

Cryo-EM is an open research area of great interest. The 3D problem is not easy to solve and there are many steps in the cryo-EM reconstruction pipeline to be further improved. With GAT-Denoiser, a neural network architecture is introduced, towards an algorithm to work for cryo-EM

GAT-Denoiser in 3D: So far, GAT-Denoiser was evaluated on a CT dataset in 2D. Therefore, the next step would be to increase the dimension and derive GAT-Denoiser to

[13] http://spr.math.princeton.edu/

work in 3D. This should be feasible without too much of an effort, as the single components of GAT-Denoiser are able to work in 3D as well.

Drop known angle assumption: So far, GAT-Denoiser works with the assumption that angles are fixed. In a future work, this assumption could be dropped. GL embedding is a way to approximate angles for CT and cryo-EM observations, but the quality in the high noise regime is rather low. To extend GAT-Denoiser to work with unknown angles, it needs to be combined with a fourth component, which can successfully approximates angles in the high noise regime.

Further cryo-EM challenges: In this Thesis, only single particle cryo-EM is considered. When GAT-Denoiser is available in 3D, one could also think of to approach the problem where from a given observation, underlying sample is not structural equivalent and there are some slight differences. One needs to approximate the distribution of underlying samples. This could be established first for known observation angles and further brought to unknown observation angles.

Bibliography

[1] Tamir Bendory, Alberto Bartesaghi, and Amit Singer. Single-particle cryo-electron microscopy: Mathematical theory, computational challenges, and opportunities. *IEEE Signal Processing Magazine*, 37(2):58–76, 2020. ISSN 1053-5888. doi: 10.1109/MSP.2019.2957822. URL http://dx.doi.org/10.1109/MSP.2019.2957822.

[2] David J Brenner and Eric J Hall. Computed tomography—an increasing source of radiation exposure. *New England journal of medicine*, 357(22):2277–2284, 2007. ISSN 00284793. doi: 10.1056/NEJMra072149. URL http://dx.doi.org/10.1056/NEJMra072149.

[3] Antoni Buades, Bartomeu Coll, and J-M Morel. A non-local algorithm for image denoising. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 60–65. IEEE, 2005. doi: 10.1109/CVPR.2005.38. URL http://dx.doi.org/10.1109/CVPR.2005.38.

[4] Lawrence Cayton. Algorithms for manifold learning. *Univ. of California at San Diego Tech. Rep*, 12(1-17):1, 2005.

[5] Rolf Clackdoyle and Michel Defrise. Tomographic reconstruction in the 21st century. *IEEE Signal Processing Magazine*, 27(4):60–80, 2010. ISSN 10535888. doi: 10.1109/MSP.2010.936743. URL http://dx.doi.org/10.1109/MSP.2010.936743.

[6] Ronald R Coifman and Stéphane Lafon. Diffusion maps. *Applied and computational harmonic analysis*, 21(1):5–30, 2006. ISSN 1063-5203. doi: 10.1016/j.acha.2006.04.006. URL http://dx.doi.org/10.1016/j.acha.2006.04.006.

[7] Kostadin Dabov, Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian. Image denoising by sparse 3-d transform-domain collaborative filtering. *IEEE Transactions on Image Processing*, 16(8):2080–2095, 2007. doi: 10.1109/TIP.2007.901238. URL http://dx.doi.org/10.1109/TIP.2007.901238.

[8] Allison Doerr. Single-particle cryo-electron microscopy. *Nature methods*, 13(1):23–23, 2016. ISSN 1548-7091. doi: 10.1038/nmeth.3700. URL http://dx.doi.org/10.1038/nmeth.3700.

[9] Yifeng Fan and Zhizhen Zhao. Multi-frequency vector diffusion maps. In *International Conference on Machine Learning*, pages 1843–1852. PMLR, 2019. doi: 10.1002/cpa.21395. URL http://dx.doi.org/10.1002/cpa.21395.

[10] Yifeng Fan and Zhizhen Zhao. Cryo-electron microscopy image denoising using multi-frequency vector diffusion maps. In *2021 IEEE International Conference on Image Processing (ICIP)*, pages 3463–3467. IEEE, 2021. ISBN 978-1-66544-115-5. doi: 10.1109/ICIP42928.2021.9506435. URL http://dx.doi.org/10.1109/ICIP42928.2021.9506435.

[11] Marco Gori, Gabriele Monfardini, and Franco Scarselli. A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pages 729–734. IEEE, 2005. doi: 10.1109/IJCNN.2005.1555942. URL http://dx.doi.org/10.1109/IJCNN.2005.1555942.

[12] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. doi: 10.48550/arXiv.1412.6980. URL https://doi.org/10.48550/arXiv.1412.6980.

[13] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[14] Johannes Leuschner, Maximilian Schmidt, Daniel Otero Baguer, and Peter Maaß. The lodopab-ct dataset: A benchmark dataset for low-dose ct reconstruction methods. *arXiv preprint arXiv:1910.01113*, 2019.

[15] Johannes Leuschner, Maximilian Schmidt, Poulami Somanya Ganguly, Vladyslav Andriiashen, Sophia Bethany Coban, Alexander Denker, Dominik Bauer, Amir Hadjifaradji, Kees Joost Batenburg, Peter Maass, et al. Quantitative comparison of deep learning-based image reconstruction methods for low-dose and sparse-angle ct applications. *Journal of Imaging*, 7(3):44, 2021. ISSN 2313-433X. doi: 10.3390/jimaging7030044. URL http://dx.doi.org/10.3390/jimaging7030044.

[16] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. ISBN 978-3-31924-573-7. doi: http://dx.doi.org/10.1007/978-3-319-24574-4_28. URL 10.1007/978-3-319-24574-4_28.

[17] Amit Singer. Mathematics for cryo-electron microscopy. In *Proceedings. of the International Congress of Mathematicians: Rio de Janeiro 2018*, pages 3995–4014. World Scientific, 2018. ISBN 978-9-81327-287-3. doi: 10.1142/9789813272880_0209. URL http://dx.doi.org/10.1142/9789813272880_0209.

[18] Amit Singer and H-T Wu. Vector diffusion maps and the connection laplacian. *Communications on pure and applied mathematics*, 65(8):1067–1144, 2012. ISSN 0010-3640. doi: 10.1002/cpa.21395. URL http://dx.doi.org/10.1002/cpa.21395.

[19] Daniel Spielman. Spectral graph theory. *Combinatorial scientific computing*, 18, 2012. doi: 10.1201/b11644-19. URL http://dx.doi.org/10.1201/b11644-19.

[20] Peter Toft. The radon transform. *Theory and Implementation (Ph. D. Dissertation)(Copenhagen: Technical University of Denmark)*, 1996.

[21] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[22] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[23] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17 (4):395–416, 2007. doi: 10.1007/s11222-007-9033-z. URL http://dx.doi.org/10.1007/s11222-007-9033-z.

[24] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*, 38(5):1–12, 2019. ISSN 07300301. doi: 10.1145/3326362. URL http://dx.doi.org/10.1145/3326362.

[25] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *International Conference on Machine Learning*, pages 6861–6871. PMLR, 2019.

A

Mathematical Tools

A.1 3D rotation matrix

A rotation matrix is a transformation matrix used to perform rotations. In 3D case, matrix for rotating one single axis can be described as:

$$R_{e_x}(\theta)\begin{bmatrix}1 & 0 & 0\\ 0 & \cos\theta & -\sin\theta\\ 0 & \sin\theta & cos\theta\end{bmatrix} \tag{A.1}$$

$$R_{e_y}(\theta)\begin{bmatrix}\cos\theta & 0 & \sin\theta\\ 0 & 1 & 0\\ -\sin\theta & 0 & cos\theta\end{bmatrix} \tag{A.2}$$

$$R_{e_z}(\theta)\begin{bmatrix}\cos\theta & -\sin\theta & \\ \sin\theta & \cos\theta & 0\\ 0 & 0 & 1\end{bmatrix} \tag{A.3}$$

Where e_x, e_y, e_z corresponds to the axis unit-vector (for x: $(1,0,0)$, etc.) and $\theta \in \mathbb{R}$. To combine the single axis rotations, matrices can be multiplied with each other:

$$R(\theta) = R_{e_x}(\theta)R_{e_y}(\theta)R_{e_z}(\theta) \tag{A.4}$$

In Equation A.4, angle θ is the same for all axis, which does not have to be.

A.2 Power Iterations

Power iteration, also called power method, is an iterative method that approximates the largest eigenvalue of a diagonalizable matrix A.

The algorithm starts with a random vector b_0 or an approximation of the dominant eigenvector.

$$b_{k+1} = \frac{Ab_k}{||Ab_k||} \tag{A.5}$$

It will not necessarily converge. It only converges if A has an eigenvalue strictly grater than its other eigenvalues and initial vector b_0 is not orthogonal to the eigenvector associated with the largest eigenvalue.

B

Graph Laplacian Embedding

B.1 Manifolds

TODO: it maps Rd to RD, if d¡D. In our case, it map an angle in S1 (subset of R) to something in RM. The circle S1, is 1 dimensional, it depends only on 1 parameter, and can be represented in R2 like you did. I thinks (not 100 of sure of that right now) that the rotation group is a manifold, it maps 1d parameter (rotation angle) to some rotation operator (which is high dimensional) we don't calculate manifold. Diffusion maps (or graph Laplacian embedding) are a way to map high dimensional dataset to low dimensional one. When the data points are sampled from a manifold, the GL embedding is closely related to the manifold itself. We can expect the GL embedding to share property of the original manifold, such as preservation of distances between points

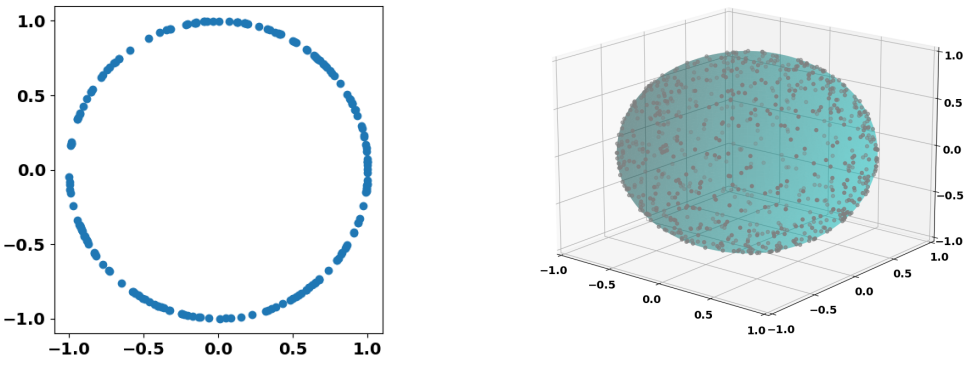
For high-dimensional data Euclidean distances are not meaningful in the sense that they will not capture similar data points well. GL can be used to compute a low-dimensional embedding which can map from high-dimensional space to low-dimensional one. In low-dimensional space, Euclidean distances make sense again.

The manifold assumption is a popular assumption for high-dimensional datasets. For a given dataset in high-dimension, one can assume that data points are samples drawn from a low-dimensional manifold, that embeds the high-dimensional space. Therefore, if underlying manifold can be approximated, a dimensionality reduction is established as one can embed data points in the low-dimensional manifold space. There is a complete area of research devoted to this manifold assumption called Manifold Learning[4].

Definition: Let manifold M be defined as $\mathcal{M} = \{f(x), f \in C^K, f : \mathbb{R}^D \to \mathbb{R}^d\}$. Manifolds are a well established mathematical concept. In this Thesis, only C^k differentiable d-dimensional manifolds defined by $\mathcal{M}$ are considered. When $d \ll D$, manifolds define a *low-dimensional embedding*, which maps from high-dimensional space $\mathbb{R}^D$ to low-dimensional space $\mathbb{R}^d$.

Let's give two popular examples of manifolds, namely the *circle* and the *sphere*. The circle is a 1D manifold, where $d = 1$ and $D = 2$. A sphere is a 2D manifold with $d = 2$ and $D = 3$. In Figure B.1(a), 200 samples are drawn from a uniform distribution of the unit-circle manifold

and in Figure B.1(b), 800 samples are drawn from a uniform distribution of the unit-sphere manifold, as well as the sphere itself.



(a) (b)

Figure B.1: Samples drawn from 1D and 2D manifold: B.1(a) circle samples, B.1(b) sphere samples

GL-manifold A simple low-dimensional embedding of a dataset can be computed with Graph-Laplacian by the following:

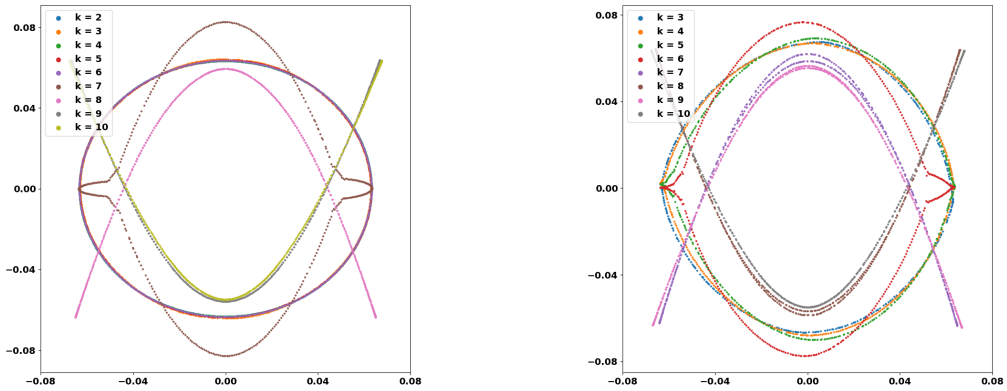
1. Construct k-NN graph from observations.
2. Calculate the (normalized) Graph Laplacian (see Equation **??**).
3. Extract the second, third (and fourth) the smallest eigenvectors.

Another popular algorithm for calculating a low-dimensional embedding is diffusion maps [6], which is a non-linear approach using Graph Laplacian. Vector diffusion maps [18] generalize the concept of diffusion maps for vector fields. Multi-frequency vector diffusion maps [9] can be seen as an extension to vector diffusion maps, which works well even on highly noisy environments. Fan and Zhao [10] successfully applied multi-frequency vector diffusion Maps in cryo-EM setting, where it was used for denoising observations.

B.1.1 Embedding quality

Finding a good embedding, which approximates the GL-manifold, is not easy and in our case, the embedding is dependent on k during graph construction as well as parameter θ, s and η for obtaining observations.

K is an important parameter for building up a graph. If set too low, neighbors do not capture similar data well as too few nodes are connected. Further, if k is set too high, strength of a neighbor is weakened and data is not well explained. In Figure B.2(a), GL-manifold computed by clean sinogram and k from 2 to 10 is illustrated. One can see, that from $k \leq 4$ GL-manifold results in a perfect circle and with $k > 4$ is moves further away from the circle.



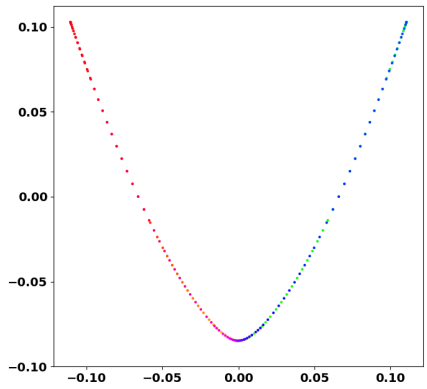
(a) Clean sinogram GL embeddings for different k.

(b) Noisy sinogram GL embeddings for different k.

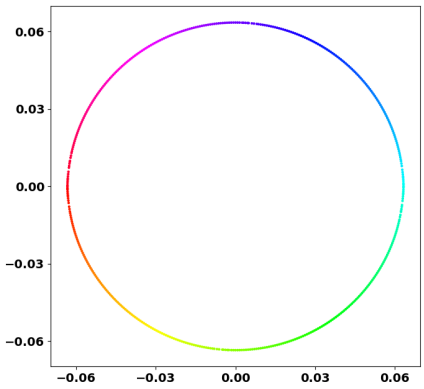
Figure B.2: Shepp-Logan phantom sinogram GL eigenvectors

If data is noisy, it is expected to be harder to construct a meaningful GL-manifold, as some connections within the graph will be noisy. This is exactly what is illustrated in Figure B.2(a), where different GL-manifold for noisy sinogram (SNR=20dB) and k from 3 to 10 are illustrated. GL-manifold can never express data with a perfect circle. As noise is chosen rather moderate, GL-manifold has still some power to express underlying data and is expected to decrease, if noise is increased.

Further, when observing a sinogram, θ defines how many observations (straight lines) are drawn and $dim(s)$ defines the amount of sampling points. Both have great impact to expressiveness of our sinogram. In Figure B.3(a) GL-manifold with $\theta \in \mathbb{R}^{200}$ and k=6 is illustrated for clean sinogram. It looks like 6 are too many neighbors, as the perfect circle cannot be established anymore. But, if θ is increased $\theta \in \mathbb{R}^{500}$, more nodes are available to choose good neighbors from and a perfect circle can be established (figure B.3(b)).



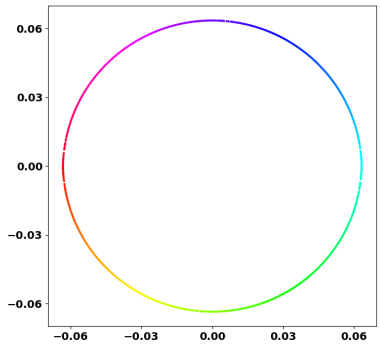
(a) Clean sinogram GL-manifold, $k = 6$ and 200 samples



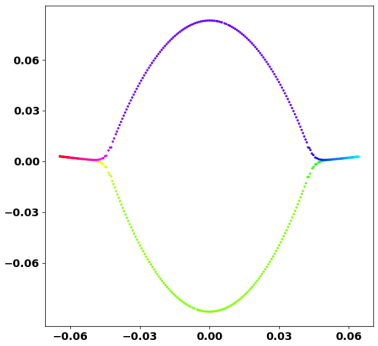
(b) Clean sinogram GL-manifold, $k = 6$ and 500 samples.

Figure B.3: Shepp-Logan phantom sinogram GL embeddings: Importance of number of samples

Moreover, the number of sampling points is important as well. For more sampling points it is expected to be harder to come up with good neighbors (fixing k and number of samples), as more data need to be explained with the same amount of neighbors, and it is more likely, that nodes are connected wrongly. This can be seen in Figure B.4(c) and Figure B.4(d), where with $dim(s) = 200$, the perfect circle can be established and with $dim(s) = 400$, not anymore (by same parameter $k = 6$ and $\theta \in \mathbb{R}^{500}$).



(c)



(d)

Figure B.4: Shepp-Logan phantom sinogram GL-manifolds: Importance of number of samples. B.4(c) Clean sinogram GL-manifold, $k = 6$ and resolution = 200, B.4(d) Clean sinogram GL-manifold, $k = 6$ and resolution = 400

Since the GL-manifold is sensitive to k, it is best practice to try different values in order to find the best GL-manifold.

C
Neural Networks

In GAT-Denoiser, different components are used, which are all neural networks. In this chapter, they are introduced in detail.

C.1 Graph Attention Networks

The main component of GAT-Denoiser is a GAT. GAT is an extension to GCN and adds attention (or weights) to neighbors for learning new node feature representations. Topology of the graph will not change but weighted averaging over the neighborhood will be computed.

Single Layer Input of a single GAT layer are node features $h = \{h_1, h_2, \dots, h_N\} \in \mathbb{R}^F$, where N is the number of nodes and F the number of features per node. Single layer will map input to output, which can potentially have different dimensions: $h' = \{h'_1, h'_2, \dots, h'_N\} \in \mathbb{R}^{F'}$ As is other GNNs, input features are initially linearly transformed and parametrized by a learnable weight matrix $W \in \mathbb{R}^{F' \times F}$. This allows to add enough expressiveness to the neural network and weights are learned during training. Further, attention coefficients are computed, which indicates importance of node j to node i:

$$e_{ij} = a(Wh_i, Wh_j) \tag{C.1}$$

With a as the shared attentional mechanism $a : \mathbb{R}^{F'} \times \mathbb{R}^{F'} \mapsto \mathbb{R}$. Veličković et al. [22] proposed to use a single-layer feedforward neural network, parametrized by a weight vector $a \in \mathbb{R}^{2F'}$ and LeakyReLu as activation function.
To compare coefficients e across different nodes, normalization is needed. Therefore, softmax is used as normalization $\alpha_{ij} = softmax_j(e_{ij})$ such that all attention coefficient of one node sum up to 1 and therefore are nicely comparable across nodes. Finally, new node embedding is calculated as:

$$h'_i = \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij} W h_j\right) \tag{C.2}$$

With σ as some arbitrary activation function.

Multi-Head attention Motivated by the work of Vaswani et al. [21], multi-head attention can be beneficial to stabilize learning process. Therefore, not only a single weight matrix is learned, but W is split up in several parts, all learned individually:

$$h_i' = \Big\Vert_{k=1}^{K} \sigma\left(\sum_{j\in\mathcal{N}_i} \alpha_{ij}^k W^k h_j\right), \tag{C.3}$$

Where $\|$ corresponds to concatenation, α_{ij}^k the k-th attention mechanism and W^k the linear transformations weight matrix. The final output consists of KF' output features.

Last layer: In the last layer, output dimension needs to be obtained. Consequently, concatenation is no longer plausible and averaging is used to match desired dimension.

C.2 Convolution

Convolution is an important part in Signal Processing, because it allows to average an incoming signal. Convolution commonly operators on pixel spaces, where every observation location or time slot gets one value assigned.

$$x \star k = y, \tag{C.4}$$

Where x is input signal, k is kernel, $\star$ the convolution operator and y the convolved signal. To apply convolution, a kernel with weights needs to be defined. This kernel will then slide over the input signal x and computes the dot product with its weights. Figure C.1 illustrates 1D convolution. Kernel size is set to 3, n refers to input signal size and m to output signal size. In the example $n = m + 2$, thus, the convolved output signal size will be decreased by 2. The concept of convolution can be extended to arbitrary dimensions.

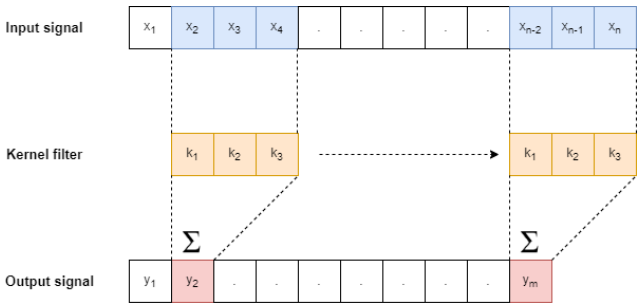


Figure C.1: 1D Convolution

Padding: Padding can be defined to add pixels to boundary of the signal. In the example, padding is set to 0 and therefore, the signal size is decreased by 2. If signal size should be fixed during convolution, padding is a powerful tool. For padding 1, the input size n would be equal to output size m, as an extra element in the output signal will be at convolved at boundaries.

Stride: Stride is the parameter how far kernel moves each time. In the example, stride was defined to be 1. If stride is increased, the output signal size will decrease.

C.3 U-Net

U-Net can boost performance of CT reconstruction, where FBP is further refined with a neural network. It is a convolutional neural network, which is well suited for image segmentation in different domains. [16] showed great success for biomedical image segmentation. The neural network architecture consists of contracting path and expansive path, resulting in a U-shape, as illustrated in Figure C.2.

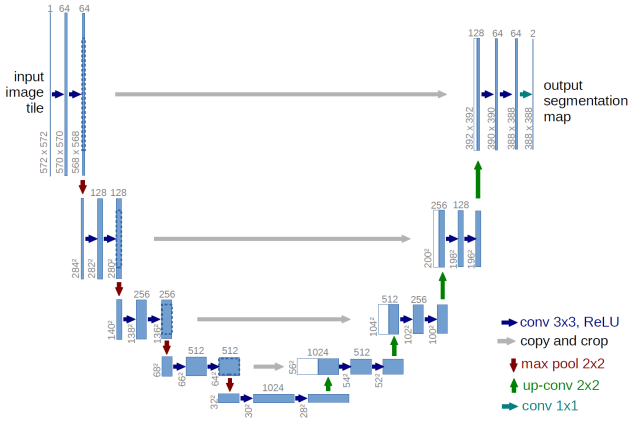


Figure C.2: U-Net architecture [16, p 2, Fig. 1]
Number on top of boxes denotes channels, where numbers at bottom of boxes refer to input dimension.

During contracting path (left part), input dimension is decreased but channels increased. For every step in the contracting path, two 3x3 convolution layers are followed by ReLu and a 2x2 max pooling for down-sampling. Further, at each down-sampling step, input channels are doubled. Multiple contracting steps are combined. After last contracting step, expansive path (right part) starts where input dimension will be increased and input channels will be decreased. For every expansive step, an up-sampling of the feature map takes place, followed by a 2x2 convolution, which halves the number of channels. Then, concatenation with the corresponding feature map of the contracting path is done (gray arrow in Figure C.2), followed by again two 3x3 convolutions and ReLu. Final layer is a 1x1 convolution to map to desired output dimension and single output channel.

D

Additional results

In this chapter, additional numeric and visual results are presented.

D.1 Small experiment: GAT-Denoiser components

Table D.1 shows numerical results for the small scale experiment. Results for the baseline algorithms, different models with $k = 2$ and $k = 8$ are presented.

Algorithm	**SNR$_y$ 0 dB SNR**	**SNR$_y$ -5 dB SNR**	**SNR$_y$ -10 dB SNR**	**SNR$_y$ -15 dB SNR**
FBP	4.50	-0.31	-5.25	-10.22
BM3D sino	9.93	7.42	4.61	2.00
BM3D reco	10.79	8.09	4.97	1.54
U-Net	7.13	6.18	4.34	1.87
K-NN k = 2				
GAT	9.42	8.25	6.78	5.75
Conv+GAT	10.01	8.88	7.69	6.79
GAT+U-Net	7.93	7.28	6.77	4.91
GAT+U-Net*	11.92	10.14	10.13	8.43
GAT+U-Net**	13.41	12.12	10.62	8.50
Conv+GAT+U-Net	8.17	7.09	4.53	5.67
Conv+GAT+U-Net*	12.60	10.71	8.80	8.44
Conv+GAT+U-Net**	12.80	11.70	10.63	9.21
K-NN k = 8				
GAT	8.32	6.94	6.19	5.01
Conv + GAT	9.05	8.07	6.99	5.75
GAT + U-Net	7.49	6.40	6.01	5.07
GAT + U-Net**	11.88	**10.85**	**9.68**	**8.26**
Conv + GAT + U-Net	7.55	6.50	5.74	4.87
Conv + GAT + U-Net**	**11.94**	10.31	7.34	7.33

The evaluated GAT-Denoiser models have been trained with 1024 images for 200 epochs. U-Net* refers to joint U-Net training and U-Net** to joint U-Net training, after a training of 100 epochs of GAT-Denoiser alone.

Table D.1: Small scale experiment: GAT-Denoiser components vs. baseline algorithms.

D.1.1 Small experiment - k-NN with 2 layers, 4 heads

D.2 Small experiment: further k-NN results

K-NN experiments have been computed before the one with GAT parameter for layers and heads. The best value for GAT parameters have been 2 layers and 4 heads. Therefore, k-NN experiment have been repeated with these settings and results can be found in Table D.2. The outcome was almost the same

k	**SNR$_y$ 0 dB**	
	Loss	**SNR**
2	9.90	601.38
4	9.64	622.13
6	9.38	634.44
8	9.19	648.97

Table D.2: Different k values for SNR$_y$ 0 dB and -10 dB

D.3 Small experiment - GAT dropout

TODO: define experimet fixed parameters.

GAT dropout.	**SNR$_y$ 0 dB**	
	Loss	**SNR**
0	8.33	739.5
0.01	7.81	766.2
0.03	7.98	743.3
0.05	8.09	738.4
0.1	7.79	761.8

Table D.3: Different k-NN values for SNR$_y$ 0 dB and -10 dB

D.4 Large scale experiment: baseline results

Table D.4 presents the result of the choosen baseline algorithms for the large scale experiment.

Algorithm	**SNR$_y$ 0 dB**		**SNR$_y$ -5 dB**		**SNR$_y$ -10 dB**		**SNR$_y$ -15 dB**	
	Loss	**SNR**	**Loss**	**SNR**	**Loss**	**SNR**	**Loss**	**SNR**
FBP	4.57	1167.6	-0.19	1973.5	-5.17	3425.3	-10.09	10'737.3
BM3D sino	9.93	709.9	7.40	890.6	4.61	1168.0	2.09	1570.0
BM3D reco	10.85	658.2	8.15	818.6	5.07	1118.3	1.76	1662.5
U-Net	7.18	968.8	6.34	1044.7	4.50	1214.1	1.94	1522.4

Table D.4: Baseline results large experiments

D.4.1 Large scale experiment: GAT-Denoiser models

Table D.5 presents the final results for baseline algorithms and GAT-Denoiser models for the large scale.

Algorithm	**SNR_y 0 dB SNR**	**SNR_y -5 dB SNR**	**SNR_y -10 dB SNR**	**SNR_y -15 dB SNR**
FBP	4.57	-0.19	-5.17	-10.09
BM3D sino	9.93	7.40	4.61	**2.09**
BM3D reco	**10.85**	**8.15**	**5.07**	1.76
U-Net	7.18	6.34	4.50	1.94
K-NN k = 2				
GAT	9.36	7.89	6.64	5.32
Conv+GAT	10.05	9.07	7.17	6.15
GAT+U-Net	8.39	7.49	6.60	5.66
GAT+U-Net**	13.34	12.07	**11.46**	**9.62**
Conv+GAT+U-Net	8.31	7.47	6.66	5.15
Conv+GAT+U-Net**	**13.43**	**12.52**	11.43	9.55

Table D.5: Large scale experiment: overall GAT-Denoiser components vs. Baseline for k-NN = 2

D.5 Wandb results

Table D.6 shows urls raw wandb results for all computed experiments on the LoDoPaB-CT dataset.

Experiments	**URL**
Small scale experiments:	
FBP	https://wandb.ai/cedric-mendelin/FBP-Validation-LoDoPaB-small
BM3D	https://wandb.ai/cedric-mendelin/BM3D-Validation-LoDoPaB-small
U-Net	https://wandb.ai/cedric-mendelin/U-Net-Validation-LoDoPaB-small
K-nn	https://wandb.ai/cedric-mendelin/LoDoPaB-Small-k-NN-and-graph-size
Convolution	https://wandb.ai/cedric-mendelin/LoDoPaB-Small-Convolution
Loss	https://wandb.ai/cedric-mendelin/LoDoPaB-Small-Loss-FBP-vs-SINO
GAT dropout	https://wandb.ai/cedric-mendelin/LoDoPaB-Small-GAT-dropout
GAT-Denoiser $k = 2$	https://wandb.ai/cedric-mendelin/LoDoPaB-Small-Components-knn-2
GAT-Denoiser $k = 8$	https://wandb.ai/cedric-mendelin/LoDoPaB-Small-Components-knn-8
Large scale experiments:	
U-NET	https://wandb.ai/cedric-mendelin/U-Net-Validation-LoDoPaB-large
FBP	https://wandb.ai/cedric-mendelin/FBP-Validation-LoDoPaB-large
BM3D	https://wandb.ai/cedric-mendelin/BM3D-Validation-LoDoPaB-large
GAT-Denoiser	https://wandb.ai/cedric-mendelin/LoDoPaB-Large-knn-2

Table D.6: Wandb results.

Average wandb run result: Average SNR and loss have been computed with a python script and exported values can be found under[14].

[14] https://github.com/cedricmendelin/master-thesis/tree/main/wandb_results

D.6 Large scale experiment: Visual Results

In this section, further visual results for the large scale experiment are presented for SNR_y 0 dB, -5 dB, -10 dB and -15 dB. Clean images are repeated at every subsection, even tough they are the same

D.6.1 SNR_y 0 dB

D.6.1.1 Clean test images:

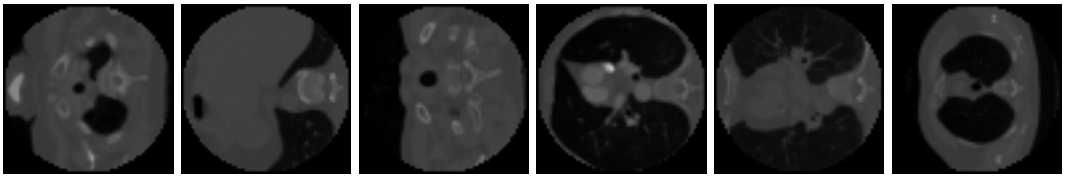


Figure D.1: Clean test images.

D.6.1.2 Baseline algorithms:

Figure D.2: FBP reconstruction for SNR_y 0 dB

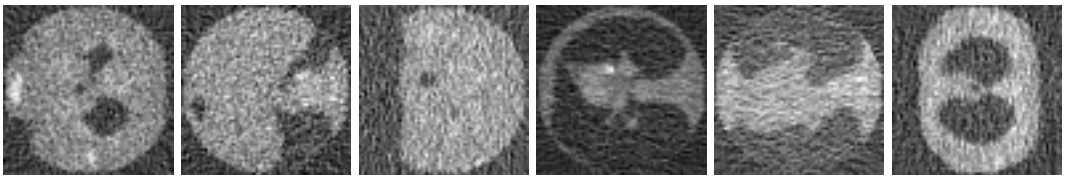


Figure D.3: BM3D sino reconstruction for SNR_y 0 dB

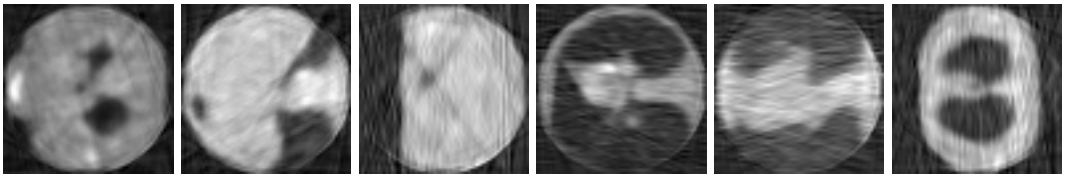


Figure D.4: BM3D reco reconstruction for SNR_y 0 dB

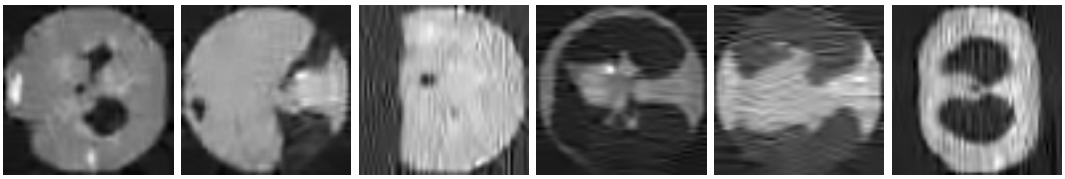
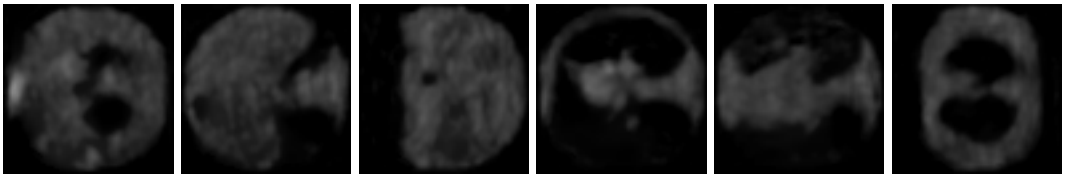


Figure D.5: U-Net reconstruction for SNR_y 0 dB



D.6.1.3 GAT-Denoiser models

Figure D.6: GAT reconstruction for SNR_y 0 dB

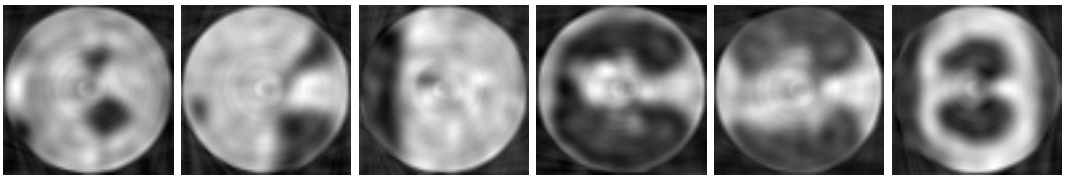


Figure D.7: Conv + GAT reconstruction for SNR_y 0 dB

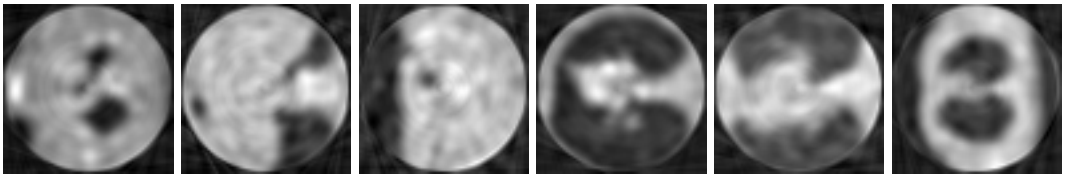


Figure D.8: GAT + U-Net reconstruction for SNR_y 0 dB

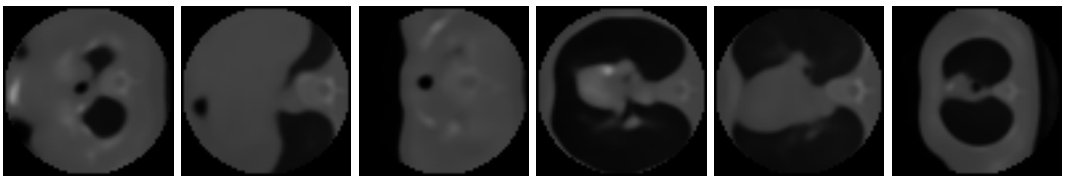
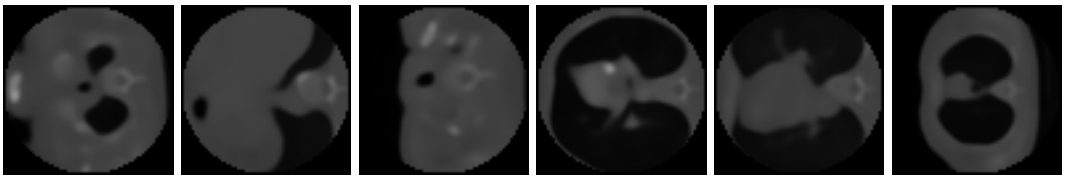


Figure D.9: Conv + GAT + U-Net reconstruction for SNR_y 0 dB



D.6.2 SNR_y -5 dB

D.6.2.1 Clean test images:

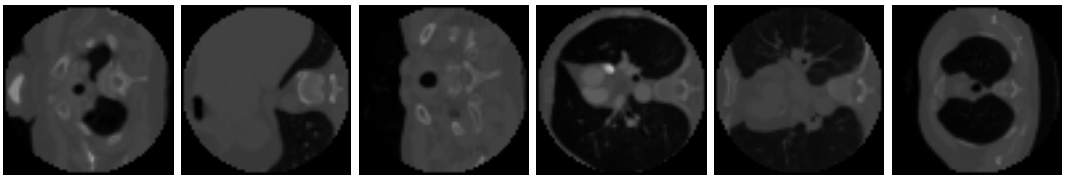


Figure D.10: Clean test images.

D.6.2.2 Baseline algorithms:

Figure D.11: FBP reconstruction for SNR_y -5 dB

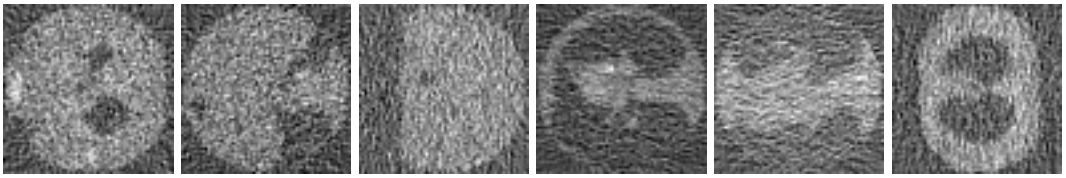


Figure D.12: BM3D sino reconstruction for SNR_y -5 dB

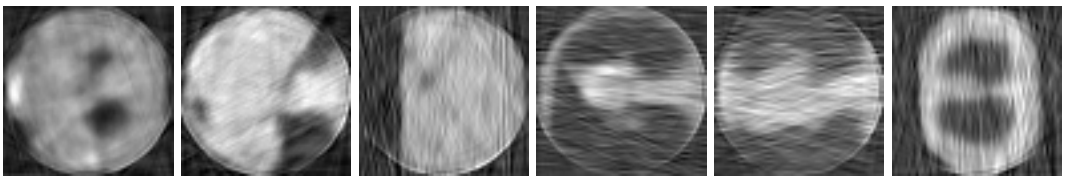


Figure D.13: BM3D reco reconstruction for SNR_y -5 dB

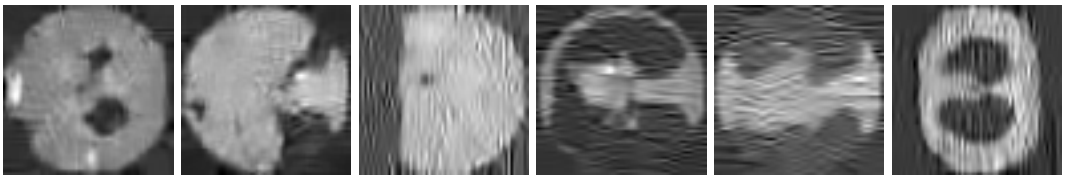
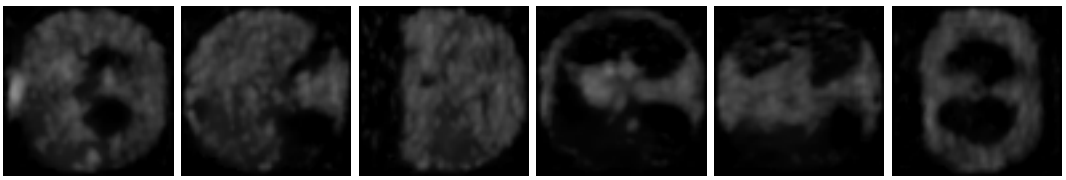


Figure D.14: U-Net reconstruction for SNR_y -5 dB



D.6.2.3 GAT-Denoiser models

Figure D.15: GAT reconstruction for SNR_y -5 dB

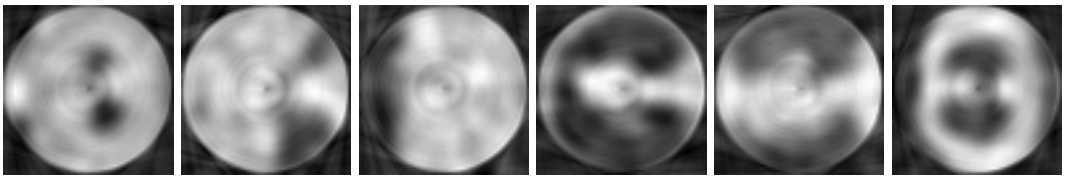


Figure D.16: Conv + GAT reconstruction for SNR_y -5 dB

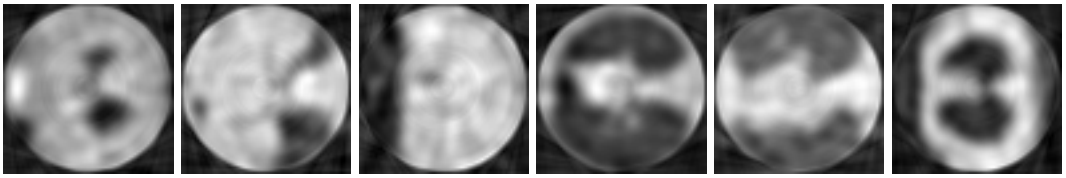


Figure D.17: GAT + U-Net reconstruction for SNR_y -5 dB

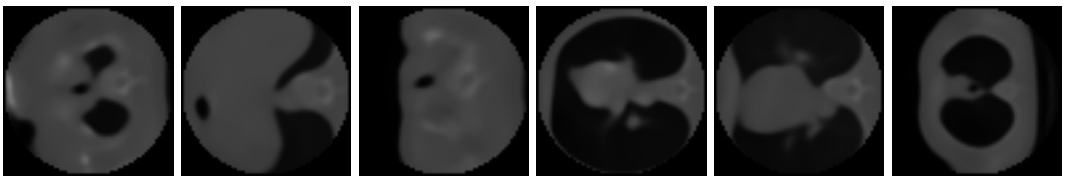
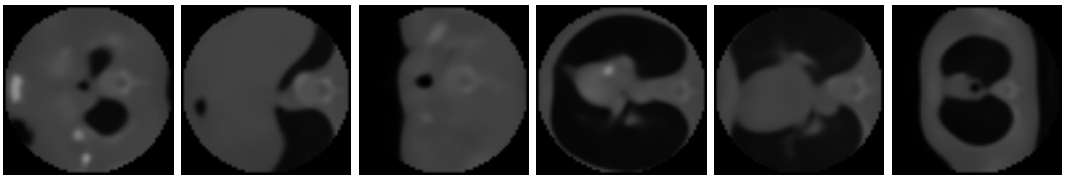


Figure D.18: Conv + GAT + U-Net reconstruction for SNR_y -5 dB



D.6.3 SNR_y -10 dB

D.6.3.1 Clean test images:

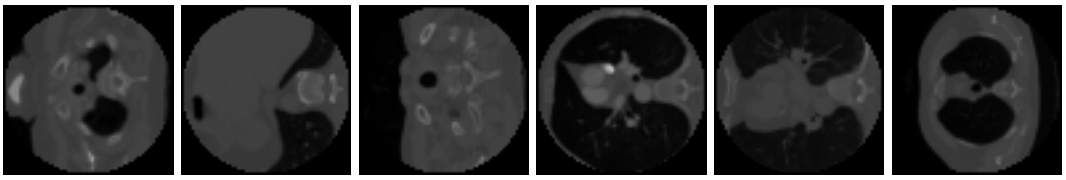


Figure D.19: Clean test images.

D.6.3.2 Baseline algorithms:

Figure D.20: FBP reconstruction for SNR_y -10 dB

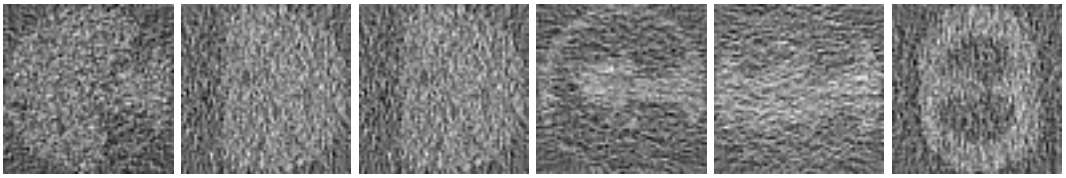


Figure D.21: BM3D sino reconstruction for SNR_y -10 dB

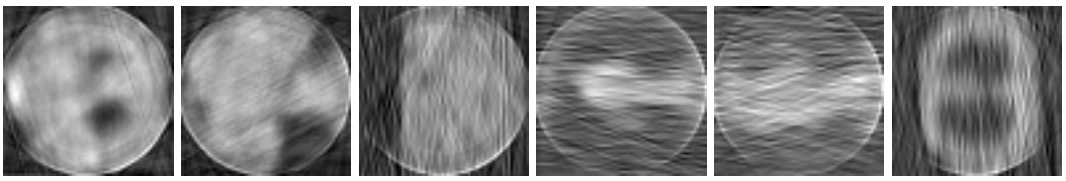


Figure D.22: BM3D reco reconstruction for SNR_y -10 dB

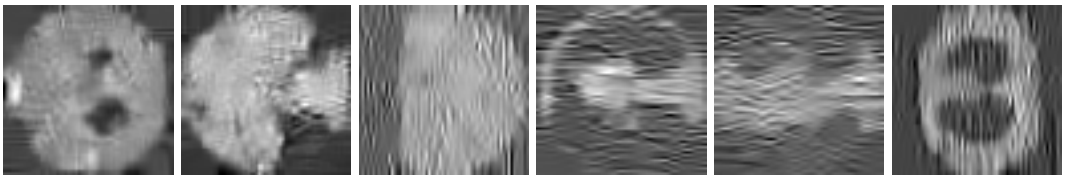
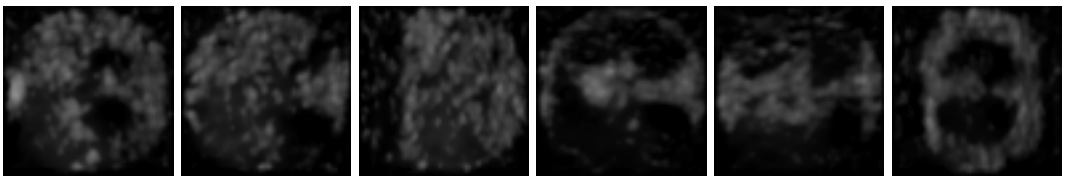


Figure D.23: U-Net reconstruction for SNR_y -10 dB



D.6.3.3 GAT-Denoiser models

Figure D.24: GAT reconstruction for SNR_y -10 dB

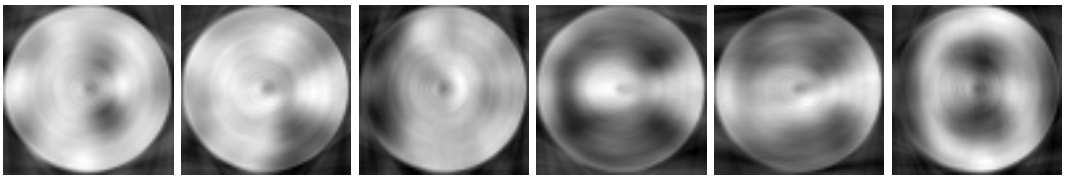


Figure D.25: Conv + GAT reconstruction for SNR_y -10 dB

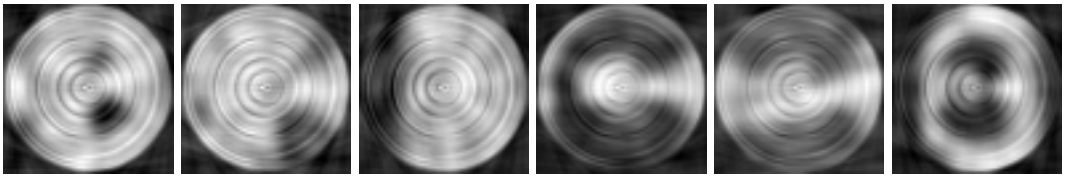


Figure D.26: GAT + U-Net reconstruction for SNR_y -10 dB

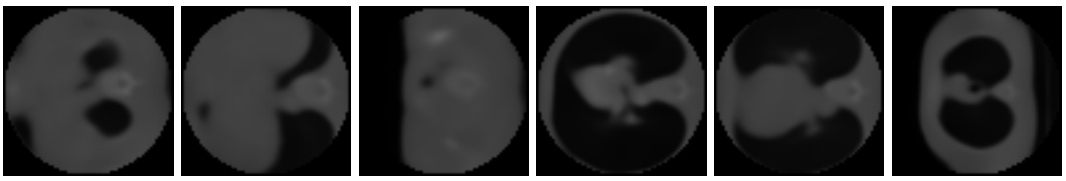
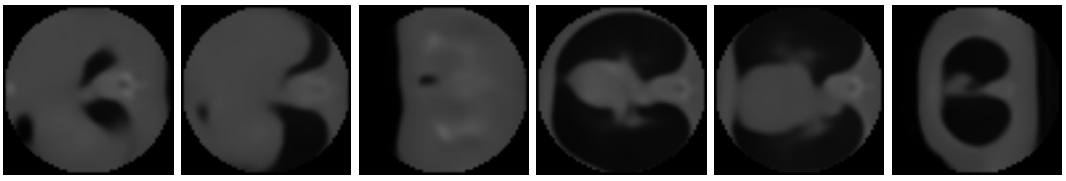


Figure D.27: Conv + GAT + U-Net reconstruction for SNR_y -10 dB



D.6.4 SNR_y -15 dB

D.6.4.1 Clean test images:

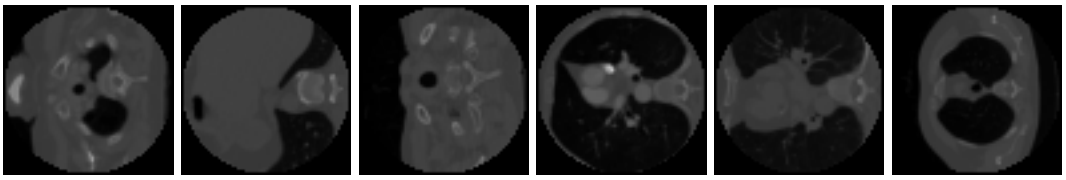


Figure D.28: Clean test images.

D.6.4.2 Baseline algorithms:

Figure D.29: FBP reconstruction for SNR_y -15 dB

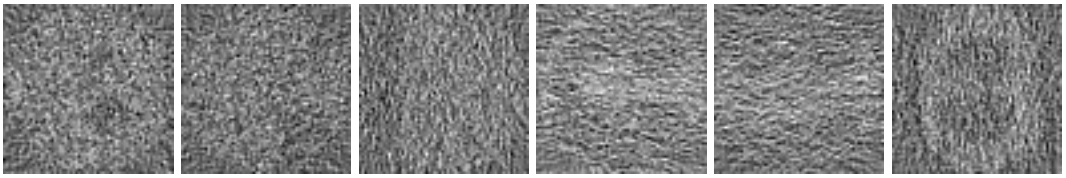


Figure D.30: BM3D sino reconstruction for SNR_y -15 dB

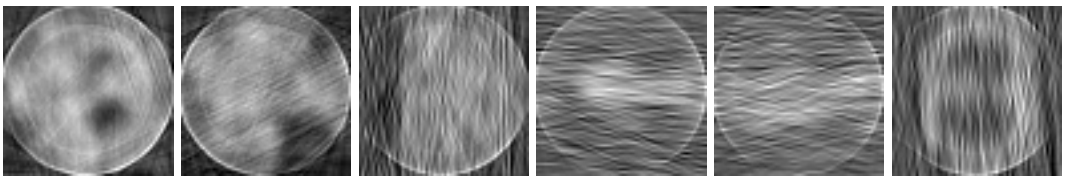


Figure D.31: BM3D reco reconstruction for SNR_y -15 dB

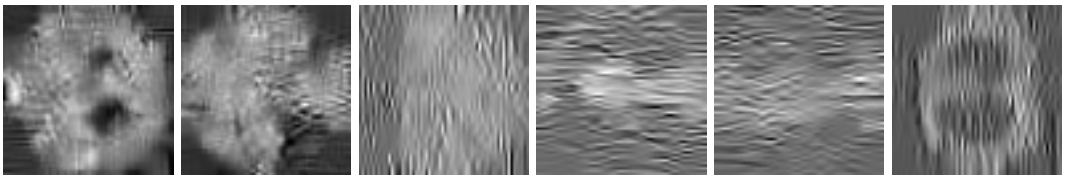
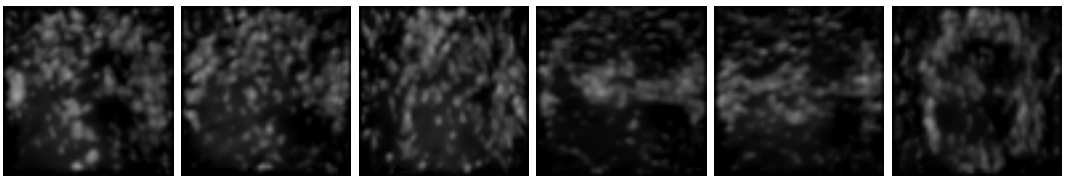


Figure D.32: U-Net reconstruction for SNR_y -15 dB



D.6.4.3 GAT-Denoiser models

Figure D.33: GAT reconstruction for SNR_y -15 dB

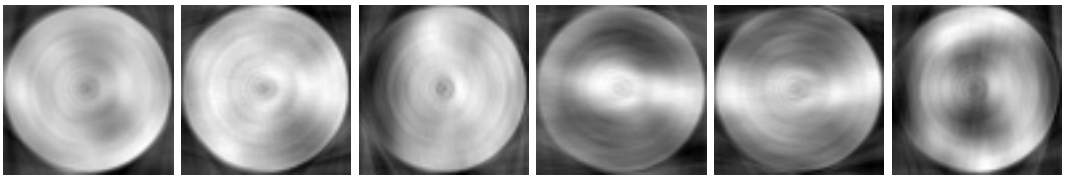


Figure D.34: Conv + GAT reconstruction for SNR_y -15 dB

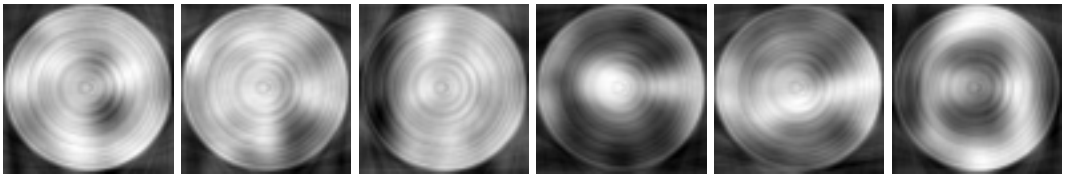


Figure D.35: GAT + U-Net reconstruction for SNR_y -15 dB

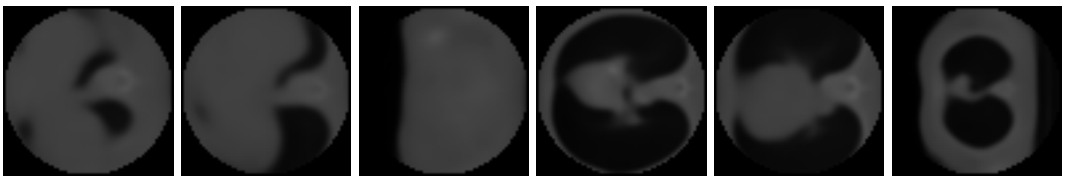
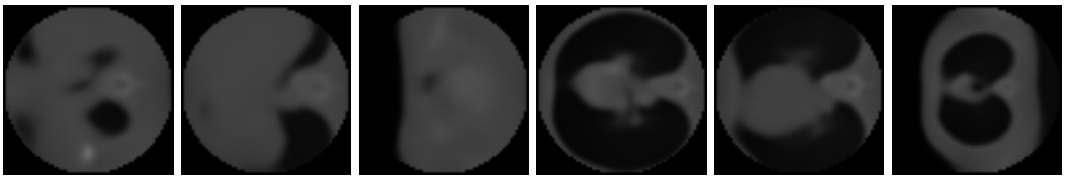


Figure D.36: Conv + GAT + U-Net reconstruction for SNR_y -15 dB



University of Basel

Faculty of Science

Declaration on Scientific Integrity

(including a Declaration on Plagiarism and Fraud)
Translation from German original

Title of Thesis:

Name Assesor: _____

Name Student: _____

Matriculation No.: _____

With my signature I declare that this submission is my own work and that I have fully acknowledged the assistance received in completing this work and that it contains no material that has not been formally acknowledged. I have mentioned all source materials used and have cited these in accordance with recognised scientific rules.

Place, Date: _____ Student: _____

Will this work be published?

☐ No

☐ Yes. With my signature I confirm that I agree to a publication of the work (print/digital) in the library, on the research database of the University of Basel and/or on the document server of the department. Likewise, I agree to the bibliographic reference in the catalog SLSP (Swiss Library Service Platform). (cross out as applicable)

Publication as of: _____

Place, Date: _____ Student: _____

Place, Date: _____ Assessor: _____

Please enclose a completed and signed copy of this declaration in your Bachelor's or Master's thesis .