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
Data-Analytics

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 Data-Analytics group.

Further, I am extremely grateful for the support by my supervisors Dr. Valentin Debarnot and Cheng Shi. We had helpful discussions throughout the project and their valuable feedback was more than welcome, helping me to finish my Master's Thesis in quality it has reached.

Additionally, many thanks to Patrick Arnold and Marc Hennemann for proofreading the Thesis, 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

TODO 100-250 words,

Table of Contents

Acknowledgments **ii**

Abstract **iii**

1 Notes **1**

1.1 ToDo 1

1.2 Questions 1

1.3 Feedback: 1

2 Introduction **2**

3 Molecular Imaging Methods **4**

3.1 Computed tomography 4

3.2 Cryo-EM 6

3.3 Abstraction 9

4 Graph Denoising **11**

4.1 Graph Foundations 11

4.1.1 Graph definition 11

4.1.2 Graph construction 12

4.2 Graph Denoising definition 13

4.3 Graph Deep Learning 14

4.4 Graph Laplacian & Manifolds 14

4.4.1 Graph Laplacian 14

4.4.2 Manifolds 15

4.4.3 Manifold assumption 16

4.4.4 GL-manifold calculation 16

4.4.5 Manifold of computed tomography and cryo-EM 16

5 GAT-Denoiser **20**

5.1 Concept 20

5.2 Components 21

5.2.1 Graph Attention Networks 22

5.2.2 Convolution 23

5.2.3 U-Net 23

5.3 Architecture 24
5.3.1 Layers 24
5.3.2 Training 25

6 Results **26**
6.1 Dataset 26
6.2 Project setup 26
6.3 Small Experiments 28
6.3.1 Baseline 28
6.3.2 Input graph structure 28
6.3.3 GAT parameters 30
6.3.4 Convolution parameters 31
6.3.5 Loss 32
6.3.6 GAT-Denoiser components 32
6.4 Large Experiments 33
6.4.1 Baseline 33
6.4.2 GAT-Denoiser components 34
6.4.3 Last two models 34

7 Conclusion and Future Work **35**
7.1 Conclusion 35
7.2 Future Work 35

Bibliography **36**

Appendix A Appendix **39**

1 Notes

1.1 ToDo

- Finalize results chapter
- Conclusion and Dicsussion
- Finalize Introduction chapter
- Abstract chapter
- Declaration on Scientific Integrity
- Appendix
- Thesis title? Somehow connect to Graph-Denoising!
- Appendix:
  - SNR
  - Activation functions ReLu, Leaky ReLu and ELu
  - Softmax
  - Erdős–Rényi graph
  - Visual Results and further calculations

1.2 Questions

Organization

- Research group name, SADA or Data-Analytics?

1.3 Feedback:

- Check section titles and subsection, no subsection x.1 but no following x.2 and further

2 Introduction

Inverse problems aim to estimate an original signal that went through a system, based on a potentially noisy output signal observations. They are widely used throughout different science directions, such as Machine Learning, signal processing, computer vision, natural language processing and others. ML is a tool to model and solve such inverse problems.

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

Cryo-electron microscopy (cryo-EM), where molecules are imaged in an electron microscope, is a molecular imaging method and gained a lot of attention in recent years. 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 many molecular structures can be observed with near-atomic resolution. The big challenge with cryo-EM is enormous noise.
TODO: Introduce CT and cryo-EM

TODO: GAT-Denoiser

TODO:Motivation and Goal

ToDo:Reference to Results

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

ToDO: Overview of chapters

3 Molecular Imaging Methods

In this Chapter, molecular imaging methods *computed tomography* and *cryo-electron microscopy* (cryo-EM) will be introduced. 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 three-dimension (3D), as original object is in 3D. Computed Tomography is a similar problem, but slightly simpler as the problem is potentially in 2D, and is therefore well suited as a first step to make an algorithm work for cryo-EM.

3.1 Computed tomography

Computed tomography (CT) is a well established molecular imaging method. Using X-ray source, fan shaped beams are produced which scan the imaging object. During scanning, many observations are collected, all taken over straight lines (for further details [2]). From these observations, original object can be reconstructed.

Tomography reconstruction: Tomographic reconstruction [5] is a popular inverse problem. The aim is to reconstruct an imaged object from observed observations. The reconstruction object can be in 2D or in 3D.

The focus in computed tomography during the Thesis will be on the 2D case, which is called *classical tomography.*

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

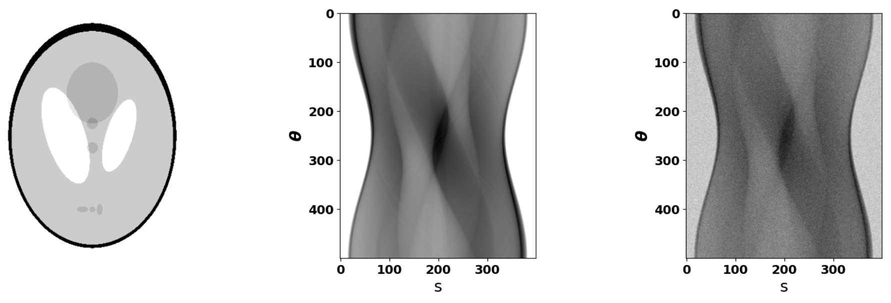
$$\begin{aligned} y_i[j] &= p_i + \eta_i[j], \text{ with } 1 \leq i \leq N \text{ and } 1 \leq j \leq M, \\ &= R(x, \theta_i, s_j) + \eta_i[j], \end{aligned} \tag{3.1}$$

with

- N: number of observations
- M: observation dimension
- $y_i \in \mathbb{R}^M$: i-th observation with $y_i[j] \in \mathbb{R}$: j-th element of observation
- $p_i \in \mathbb{R}^M$: i-th noiseless observation with $p_i[j] \in \mathbb{R}$: j-th element of noiseless observation
- $x \in L^2(\Omega)$: original object with $\Omega \subset \mathbb{R}^2$ and L^2: Lebesgue space
- $R(\cdot;\theta,s) : L^2(\Omega) \to L^2(\tilde{\Omega}), x \mapsto R(x;\theta,s)$: Radon Transform [19] with, $\tilde{\Omega} \subset \mathbb{R}$, $\theta_i \in \mathbb{R}$: observation angle and $s_j \in \mathbb{R}$: sampling point
- $\eta_i \in \mathbb{R}^M$: Gaussian noise with $\eta_i[j] \sim \mathcal{N}(0, \sigma^2) \in \mathbb{R}$

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 original object.

Observation illustration: Output of Radon Transform is called a *sinogram*, which is the CT observation. In Figure 3.1(a) the Shepp-Logan phantom can be seen. It is often used as an image for simulating a brain CT. Further, in Figure 3.1(b) and Figure 3.1(c) observation sinograms can be seen with and without noise respectively. To apply Radon Transform, parameter θ and s need to be specified. In this case, $\theta \in \mathbb{R}^{500}$ was evenly spaced between $[0, 2\pi]$ and $dim(s) = 400$. With these parameters, $p \in \mathbb{R}^{500\times400}$ and p can be plotted as image with resolution 500x400. Further, noise was added to reach a signal-to-noise-ratio (SNR) of 10dB.



(a) (b) (c)

Figure 3.1: Shepp-Logan phantom and sinograms: 3.1(a) Shepp-Logan phantom, 3.1(b) clean sinogram: $R(x, \theta, s)$, 3.1(c) noisy sinogram: $R(x, \theta, s) + \eta = y + \eta$

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

$$FBP(\cdot;\theta,s) : L^2(\tilde{\Omega}) \to L^2(\Omega), y \mapsto FBP(y;\theta,s) \tag{3.2}$$

with θ as projection angles and s sampling points. The algorithm fails when working with noisy data [16], as it is not possible anymore to draw meaningful connections.

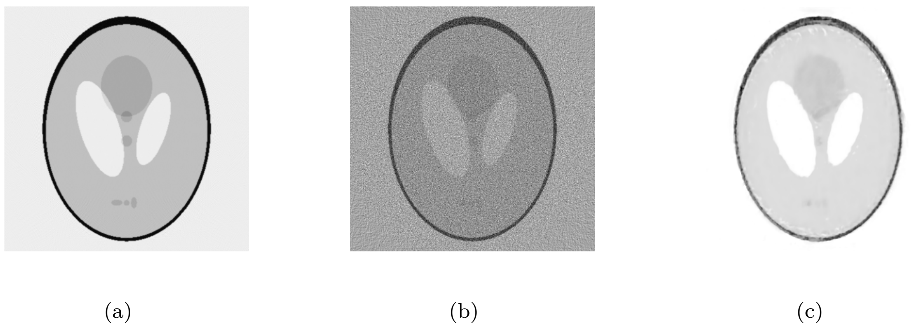


Figure 3.2: Shepp-Logan FBP reconstructions: 3.2(a) clean reconstruction: $FBP(p,\theta,s)$, 3.2(b) noisy reconstruction: $FBP(y,\theta,s)$, 3.2(c) noisy reconstruction: $UNet(FBP(y,\theta,s))$

In Figure 3.2(a) and Figure 3.2(b) reconstruction can be seen from Shepp-Logan phantom sinogram (Figure 3.1(b)) and its noisy version (Figure 3.1(c)). 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.

U-Net Leuschner et al. [14] compared different reconstruction methods for computed tomography. Today's state-of-the-art reconstruction algorithms are Deep-Learning based. U-Net [15], a convolution neural network approach, performed much better than only applying FBP, especially when dealing with noise.
During practical part of the Thesis, U-Net was used and trained on lodopab-ct dataset [13], a brain CT dataset. More details on U-Net and how it was used can be found in Chapter 5 *GAT-Denoiser* and Chapter 6 *Results*. In Figure 3.2(c), reconstruction using FBP and U-Net can be seen. Quality of reconstruction is still not perfect, but overall noise is drastically decreased.

3.2 Cryo-EM

Cryo-EM is another molecular imaging method, that enables the view of molecules in near-atomic resolution. In this Master's 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 imaging process molecules are frozen in a thin layer of ice, where they are randomly oriented and positioned. Random orientation and positioning makes reconstruction chal-

lenging, but freezing allows observation in a stable state where molecules are not moving. With an electron microscope, two-dimensional tomographic projection images of molecules in the ice 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 images. The resulting (SNR) is typically smaller than 1, which indicates that there is more noise than signal [16].
Further, observed molecules are not equal in the sense that there are some structural varieties between molecules (isotopes). While observing same molecule in ice many times, single observation could be from different isotopes.

3D cryo-EM reconstruction: Similar to tomographic reconstruction, cryo-EM reconstruction problem [1] is defined. 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, collected by the electron microscope, original object x will be estimated.
Cryo-EM reconstruction is computational intensive and multiple steps are needed to get from observed raw data to the final structure (for further details [8]).

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

$$\begin{aligned} y_i &= p_i + \eta_i, \text{ with } 1 \leq i \leq N, \\ y_i &= \Pi_z(\ \mathit{Rot}(\ x;\theta_i)) + \eta_i, \text{ with } 1 \leq i \leq N, \end{aligned} \tag{3.3}$$

where

- N: number of observations
- M: observation dimension
- $y_i \in \mathbb{R}^M$: i-th observation with $y_i[j] \in \mathbb{R}$: j-th element of observation
- $p_i \in \mathbb{R}^M$: i-th noiseless observation with $p_i[j] \in \mathbb{R}$: j-th element of observation
- $x \in L^2(\Omega)$: 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
  $R_{\theta_i} = R_{e_x}(\theta_i^{(1)}) R_{e_y}(\theta_i^{(2)}) R_{e_z}(\theta_i^{(3)}) = [R^1_{\theta_i}, R^2_{\theta_i}, R^3_{\theta_i}] \in SO(3)$
  (see A.1.3 for further details)
- $\mathit{Rot} : L^2(\Omega) \to L^2(\Omega), \mathit{Rot}(x,\theta_i) = \big((x_1,x_2,x_3) \mapsto x(x_1R^1_{\theta_i}, x_2R^2_{\theta_i}, x_3R^3_{\theta_i})\big)$: rotation operator
- $\eta_i \in \mathbb{R}^M$: Gaussian noise with $\eta_i[j] \sim \mathcal{N}(0,\sigma^2) \in \mathbb{R}$

As y_i is not observable directly, discretization is needed:

$$\begin{aligned} y_i &= (\Pi_z(\ \mathit{Rot}(\ x;\theta_i)) + \eta_i)\,(\Delta), \text{ with } 1 \leq i \leq N \\ y_i[j,k] &= \Pi_z(\ \mathit{Rot}(\ x;\theta_i))_{j,k} + \eta_i[j,k], \text{ with } 1 \leq i \leq N \text{ and } 1 \leq j,k \leq M, \end{aligned} \tag{3.4}$$

where

- $\Delta \subset \tilde{\Omega}^{M^2}$: sampling grid with dimension M^2
- Further, $y[j,k]$, $\eta[j,k]$ and $\Pi_z(\cdot)_{j,k} \in \mathbb{R}$ with j,k as indices of the sampling grid.

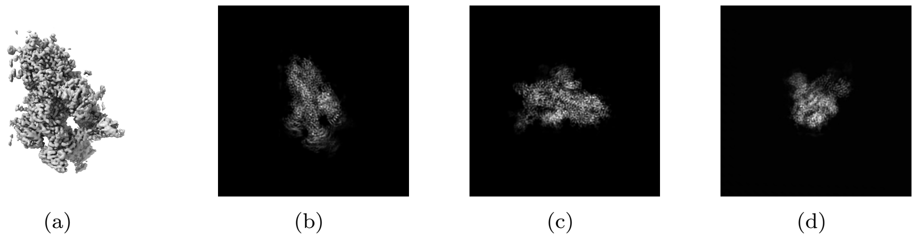


Figure 3.3: Cryo-EM reconstruction and clean projections of COVID-19 Omicron spike [2]: 3.3(a) COVID-19 Omicron spike, 3.3(b) projection along x-axis, 3.3(c) projection along y-axis, 3.3(d) projection along z-axis

Extended formula: Equation 3.3 is a simplified version of cryo-EM. First, point spread function (PSF) of the microscope is not taken into account. Secondly, structural variety is ignored, the underlying object x is not the same for every observation as modelled in the Equation. Precisely, x can be seen as a random signal from an unknown distribution defined over all possible molecules structures.
Equation can be extended and defined as the following:

$$y_i = h_i \circ \Pi_z(Rot(x_i; \theta_i)) + \eta_i, \text{ with } 1 \leq i \leq N \tag{3.5}$$

where h_i is the PSF of the microscope and $\circ$ defines the convolution. Further, $x_i \in X$ where X is the set of all possible molecule structures.

Difference to tomographic reconstruction: The problems are highly related, but cryo-EM reconstruct is more challenging. While CT observation, patient is asked to not move and therefore, angles of projections are known. Whereas, in cryo-EM, this information will be lost during freezing. Secondly, high level of noise makes cryo-EM much more challenging.

3.3 Abstraction

As tomographic reconstruction and cryo-EM reconstruction are rather similar, goal of this Thesis will be to design an algorithm, that can be applied in both scenarios.
Therefore, an abstract form of the problems 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 $\tilde{\Omega} \subset \mathbb{R}^{D-1}$.

$$\begin{aligned} y_i &= p_i + \eta_i(\Delta), \text{ with } 1 \leq i \leq N \\ y_i &= (A(x, \theta_i) + \eta_i)(\Delta), \text{ with } 1 \leq i \leq N \end{aligned} \tag{3.6}$$

with

- N: number of observations
- M: observation dimension
- $y_i \in \tilde{\Omega}^M$: the i-th observation
- $p_i \in \tilde{\Omega}^M$: the i-th noiseless observation
- $x \in L^2(\Omega)$: original object
- $A : L^2(\Omega) \to L^2(\tilde{\Omega}), x \mapsto A(x; \theta_i)$: a non-linear operator
- $\theta_i \in \mathbb{R}^P$: projection angle vector, with P projection dimension
- $\eta \sim \mathcal{N}(O, \sigma^2 I) \in \tilde{\Omega}^M$: Gaussian noise
- $\Delta \subset \tilde{\Omega}^M$: term for discretization

Reconstruction: Further, an abstract form of the reconstruction operator is defined as

$$Recon : L^2(\tilde{\Omega}) \to L^2(\Omega), y \mapsto Recon(y; \theta)(\Delta) \tag{3.7}$$

with

- $\theta_i \in \mathbb{R}^P$: projection angle vector, with P projection dimension
- $\Delta \subset \tilde{\Omega}^D$: term for discretization, D as the dimension of the space of original object

Check: Theta and discretization term

Classical tomography: Classical tomography parameters are defined with $D = 2$, $P = 1$. Further, $A(\cdot)$ is the Radon Transform (see Equation 3.1). Reconstruction operator can be defined as FBP (with or without U-Net[15]).

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. But how can it be achieved?
One idea would be to measure distances between observation. Another way is to find a low-dimensional embedding which maps our observations y to some θ. When talking from low-dimensional embeddings, there is no way around Graph Learning, which will be introduced in the following Chapter 4 *Graph Denoising*.

4 Graph Denoising

The following Chapter establishes connection between graphs and denoising in high-noise domains, such as cryo-EM. First, a broad definition of graphs is given and further, the term ”Graph Denoising” is introduced and explained. Finally, link to Graph Laplacian, Manifolds and molecular imaging methods is established.

4.1 Graph Foundations

Real world data can be in graph structure, like social networks, citation networks, protein interaction networks or a simple google search. If data is not available in graph structure, a graph can be artificially constructed with methods like k-nearest neighbors (k-NN) or others. A general framework for graph construction is introduced in Section 4.1.2 *Graph construction.*

Graph Learning: Graph Learning is a prominent research area and got a lot of attention in recent years. It is a way of applying ML on graphs and algorithms that have emerged emerged from ML but also other fields. When a graph is available, one can start using Graph Learning algorithms for solving tasks. Popular 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*, where the aim is to identify cluster 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.

4.1.1 Graph definition

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 one, an edge connects explicitly from one node to another, which means that edge $(i, j) \neq (j, i)$. In undirected graphs, edges have no direction and ordering does not matter, therefore $(i, j) = (j, i)$.
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 importance to neighbors, resulting in a *weighted* graph. *Degree* of a node are the number of incoming edges.

Adjacency matrix: To do calculations with graphs, it is common to translate graphs in a matrix, such as the adjacency matrix. The (binary) adjacency matrix of graph $G = \langle V, E\rangle$ is defined as follows:

$$A_{ij} = \begin{cases} 1 & \text{if } (i, j) \in E \\ 0, & \text{otherwise} \end{cases} \tag{4.1}$$

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, but could easily be extended by assigned not just values of 1 or 0. When G is undirected, corresponding adjacency matrix will be symmetric. Eigenvalues of A are called *spectrum* of G.

4.1.2 Graph construction

When data is not available as a graph, it can be easily constructed. Consider data from space $\Omega \subset \mathbb{R}^M$, but could basically be any arbitrary space. Then, each node is associated with some element $v \in \Omega$. Further, graph G can be constructed by using:

$$A_{ij} = \begin{cases} 1 & \text{if } dist(v_i, v_j) < \tau \\ 0, & \text{otherwise} \end{cases} \tag{4.2}$$

where v_i, v_j are nodes from indices i and j, $dist$ corresponds to a similarity (or distance) measure and τ is a threshold, when to consider two nodes to be adjacent. K-NN is one possible implementation of a graph construction algorithm, where for every node, k neighbors will be defined. The neighborhood $\mathcal{N}_i$ of v_i is defined as k nodes with the smallest similarity measure.

Noisy observations: But what happens if our data is noisy? Consider measurements which will give access to $y = p + \eta$, where $y, p \in \Omega$ and noise η is assumed to be drawn from Gaussian distribution $\mathcal{N} \sim (0, \sigma^2)$. y refers to the noisy observation and p to noiseless observation. While constructing a graph, nodes are associated with elements from p $(dist(p_i, p_j))$ in the noiseless case and from y $(dist(y_i, y_j))$ is the noisy case. Therefore, k-NN applied on noisy data, is assumed to connect some nodes wrongly, as distance measure $dist$ is not accurate due to noise. It is expected that the more noise is available in data, the more nodes are wrongly connected with k-NN.

Constructing Graphs for molecular-imaging: For a molecular-imaging (cryo-EM or computed tomography) observation, a graph can be constructed as well. Every observation y_i can be assigned to a node v_i, which means that $V \subset \mathbb{R}^M$, where M is again observation dimension. Further, $|V| = N$ where N is the number of observations, which defines how many projections from different angles θ are within the observation.
Finally, to apply Equation 4.2 similarity measure *dist* needs to be defined. For computed tomography, a distance measure between observations can be set up by using the $\ell 2$-norm $\|y_i - y_j\|$. For cryo-EM it is more challenging, as projections are drawn with some random 3D rotation and projection, it can happen that two samples are equivalent up to 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-plan rotation is added to the $\ell 2$-norm: $min_{g \in SO(1)} \|g\ y_i - y_j\|$, which is inspired by [9].
Now, everything is available to create a graph from a molecular-imaging observation.

4.2 Graph Denoising definition

First, *Graph Denoising* is not a common term in literature. In previous Section, graphs based on noisy observations were introduced and goal is to denoise these graphs, which means to estimate original graph G from a given noisy graph G_0. This is our 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 locally, by the calculus of variations or in the frequency domain [3].

Noisy Graph: 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.
Graph Denoising can therefore be written as

$$GD : A_0 \mapsto \tilde{A} \approx A \tag{4.3}$$

where A_0, $\tilde{A}$, A denotes adjacency matrix from noisy input graph, denoised graph and original graph respectively.

Connection to link prediction: Link prediction is a 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 $M_p : E' \rightarrow [0, 1]$ where E' is the set of potential links.
Further, U determines the set of all possible vertices 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 $E' \subseteq U$.

One could also say that link prediction problems are a subset of graph denoising problems.

Non-local means: In the following Section, a short introduction to the state-of-the-art image denoising method non-local means (NLM) [3] is given . For a given noisy image v, the denoised image is defined as $NL[v](i) = \sum w(i,j)\ v(j)$, where $w(i,j)$ is the weight between pixel i and j. Weights can be seen as a similarity measure of pixels, which are calculated over square neighborhoods. Similar pixel neighborhoods have a large weight and different neighborhoods have a small weight. More general, denoised image pixel i is computed as a weighted average of all pixels in the image, therefore, in a non-local way.
Non-local means is not a denoising algorithm, which works with graph as a data structure. But, it uses a neighborhood for averaging, which shows great potential for graph as a data structure for denoising, as graphs can represent neighborhoods really well.

4.3 Graph Deep Learning

As already mentioned, Graph Denoising can be seen as a way of link predication. The state-of-the-art method for solving link prediction are *Graph Deep Learning* approaches. Graph Deep Learning is a fast evolving field in research. With Graph Neural Networks (GNN) [11] the framework for neural networks with graphs has been established.
Using Graph Convolutional Networks (GCN) [12] 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 node itself in averaging, Kipf and Welling [12] 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 learning process.
Veličković et al. [21] extended the concept of GCN with attention and not all neighboring nodes get the same weight (attention). Simple Graph Convolutional Network [24] 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 (see A.1.4 *Power Iterations* for further information) over the adjacency matrix with normalization in every layer. Wang et al. [23] proposed an extension to GCN by not operating on the same graph in every layer but adopting underlying graph topology layer by layer.

4.4 Graph Laplacian & Manifolds

4.4.1 Graph Laplacian

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 and overview can

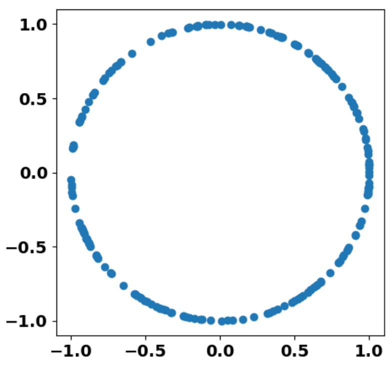
be found by [18, 22].
GL is defined as follows:

$$L = D - A, \tag{4.4}$$

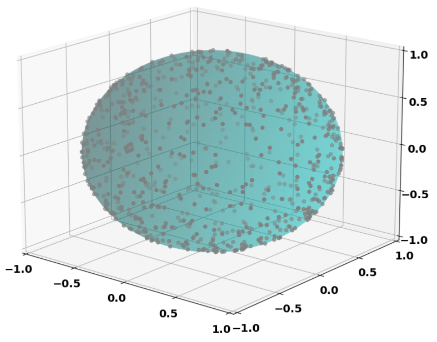
where A is the adjacency matrix and D the degree matrix (diagonal matrix with degree of nodes as entries).

4.4.2 Manifolds

In high-dimensional data Euclidean distances are not meaningful in the sense that they will not capture similar data points well. One is looking for a geodesic distance, representing the shortest path of two point on a surface. Graph Laplacian can be used to compute a *Manifold*, which can help in such scenarios. In manifold space, Euclidean distances make sense again. 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 the Master's 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 *circle* and *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 4.1(a), 200 samples are drawn from a uniform distribution of unit-circle manifold and in Figure 4.1(b), 800 samples are drawn from a uniform distribution of unit-sphere manifold, as well as the sphere itself.



(a)



(b)

Figure 4.1: Samples drawn from 1D and 2D manifold: 4.1(a) circle samples, 4.1(b) sphere samples

One popular algorithm for calculating manifolds is diffusion maps [6], which is a non-linear approach for calculating low-dimensional manifolds for (high-dimensional) datasets, using Graph Laplacian. Vector diffusion maps [17] 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 purpose.

4.4.3 Manifold assumption

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].

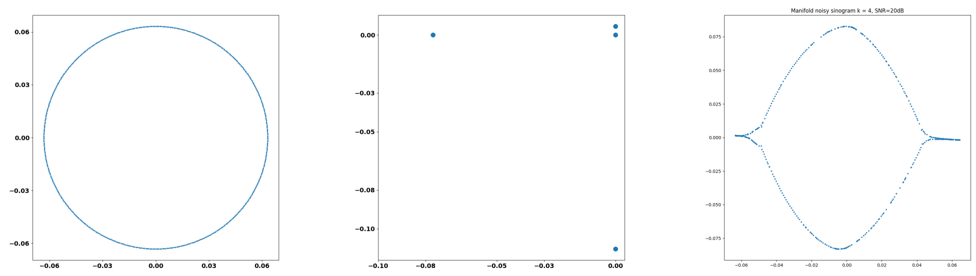
4.4.4 GL-manifold calculation

A simple low-dimensional embedding (manifold) of a dataset can be calculated with Graph-Laplacian by the following:

1. Construct k-NN graph from observations (see Section 4.1.2 *Graph construction*).
2. Calculate the (normalized) Graph Laplacian (see Equation 4.4).
3. Extract the second, third (and fourth) the smallest eigenvectors.

4.4.5 Manifold of computed tomography and cryo-EM

As a tool for GL-manifold calculation is established, it can be observed how the GL-manifold of classical tomography and cryo-EM objects look like. In the following, Shepp-Logan phantom is again used as an example of classical tomography and the low-dimensional embedding is calculated from sinogram, following instructions from Section 4.4.4 *GL-manifold calculation.*
Radon Transform, parameters θ and s where specified as $\theta \in \mathbb{R}^{500}$ as evenly spaced between $[0, 2\pi]$ and $dim(s) = 200$.
In Figure 4.2(a), GL-manifold calculated from clean sinogram and $k = 2$ can be seen. GL-manifold looks like a perfect circle. Further, noise was added to the sinogram to reach SNR 20dB and GL-manifold was computed with $k = 2$, which failed (figure 4.2(b)). But, if neighborhood is not too strictly defined and k is increased to 4, GL-manifold looks more circle like again (figure 4.2(c)).

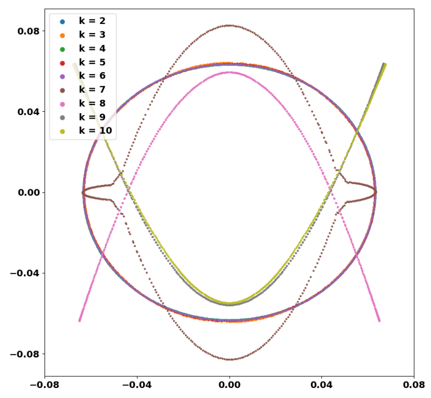


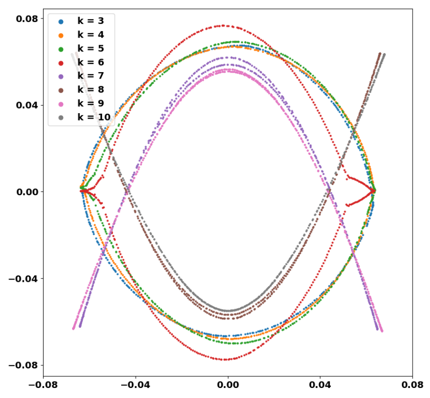
(a) (b) (c)

Figure 4.2: Shepp-Logan phantom sinogram manifolds: 4.2(a) clean sinogram manifold $k = 2$, 4.2(b) noisy sinogram manifold $k = 2$, 4.2(c) noisy sinogram manifold $k = 4$

In the field of classical tomography and cryo-EM, underlying low-dimensional GL-manifold is well-defined for none-noisy data. In the 2D case of classical tomography, underlying GL-manifold is a circle, whereas in 3D case of cryo-EM GL-manifold is defined as a sphere. This fact can be exploited during learning.

Manifold quality: Finding a good GL-manifold is not easy and in our case, GL-manifold is dependent on parameter k during graph construction as well as parameter θ, s and η for obtaining the sinogram.
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 as well. In Figure 4.3(a), GL-manifold computed by clean sinogram and k from 2 to 10 is illustrated. One can see, that from $k <= 4$ GL-manifold results in a perfect circle and with $k > 4$ is moves further away from the circle.

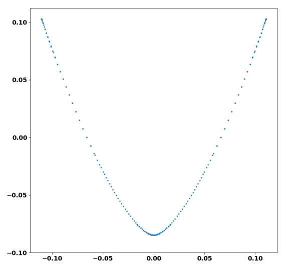


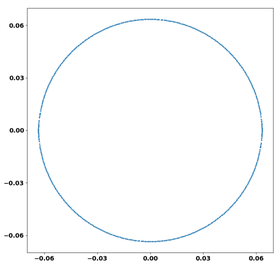


(a) (b)

Figure 4.3: Shepp-Logan phantom sinogram manifolds: 4.3(a) clean sinogram GL-manifolds, 4.3(b) noisy sinogram GL-manifolds

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 4.3(a), where different GL-manifold for noisy sinogram (SNR=20dB) and k from 3 to 10 are illustrated. GL-manifold can never express the true 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, parameter 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.





(a) (b)

Figure 4.4: Shepp-Logan phantom sinogram GL-manifolds: Importance of number of samples. 4.4(a) clean sinogram GL-manifold, $k = 6$ and 200 samples, 4.4(b) clean sinogram GL-manifold, $k = 6$ and 500 samples

In Figure 4.4(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 4.4(b)).

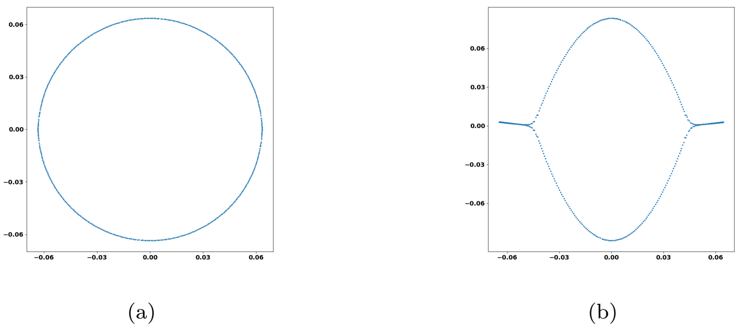


Figure 4.5: Shepp-Logan phantom sinogram GL-manifolds: Importance of number of samples. 4.5(a) Clean sinogram GL-manifold, $k = 6$ and resolution = 200, 4.5(b) Clean sinogram GL-manifold, $k = 6$ and resolution = 400

Moreover, the number of sampling points is important as well. Here, 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 4.5(a) and Figure 4.5(b), where with $dim(s) = 200$, the perfect circle can be established and with $dim(s) = 400$, not anymore (by same parameter $k = 6$ and $\theta \in \mathbb{R}^{500}$).

As with the last plots illustrated, the GL-manifold is pretty sensitive. Therefore, it is best practice to try different parameter k for finding the best GL-manifold.

TODO: Connection to next chapter

5 GAT-Denoiser

In this Chapter, contribution of this Master's Thesis is presented. As a result, a GNN can be presented which is called *GAT-Denoiser*. Its main components and overall architecture is introduced. The focus during practical part was on classical computed tomography in 2D but can be generalized to cryo-EM and 3D.

Goal: As introduced in Chapter 3 *Molecular Imaging Methods*, molecular imaging methods computed tomography and cryo-EM are the problems to approach. Further, high-noise regime is the domain of interest, to be more precise SNR between $[-10, 0]$. For a given set of observations (many sinograms or micrographs), a GNN will be trained, such that it enables denoising of observations which are expected to have a better reconstruction result after denoising. Trained model allows to denoise not seen observations which are from the same family of observations.

To start solving the problem, an algorithm was designed to work with computed tomography, additionally projection angles are assumed to be known and θ was defined as equally spaced in the interval $[0, 2\pi]$.

Input graph: To recap from Section 4.1.2 *Graph construction*, a graph constructed from molecular-imaging observation will have single observations as nodes (one horizontal line of the sinogram). As θ is fixed, angle corresponding to each single observation are known. Based on these angles, neighboring nodes can be connected.

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

5.1 Concept

In the following Section, the concept of GAT-Denoiser is introduced. GAT-Denoiser is a GNN and has two main components, namely convolution layers and GAT layers. The main

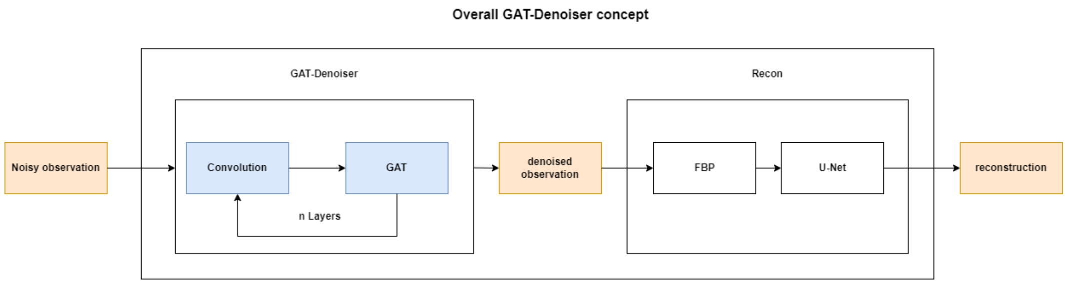


Figure 5.1: GAT-Denoiser concept

idea of GAT-Denoiser is to enable denoising of observations:

$$\textit{GAT-Denoiser}(\cdot) : L^2(\tilde{\Omega}) \to L^2(\tilde{\Omega}), y \mapsto \textit{GAT-Denoiser}(y) \tag{5.1}$$

Input y of GAT-Denoiser is a noisy observation, output will be a denoised version of it. GAT will be responsible to average over observation neighbors and convolution to denoise single observations. For every GAT layer there is a preceding convolution. In the case of computed tomography, this will be 1D convolution and for cryo-EM a 2D convolution.

The GAT is expected to 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.

$$\begin{aligned} x \approx & \textit{Recon}\,(\textit{GAT-Denoiser}\,(y))\,, \\ \text{with } & \textit{Recon} : \textit{UNet}\,(\textit{FBP}\,(\cdot)) \end{aligned} \tag{5.2}$$

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.
In Figure 5.1 overall GAT-Denoiser concept is illustrated.

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.

5.2 Components

In the following, the three main components are explained individually and in Section 5.3 *Architecture* connected to GAT-Denoiser architecture in more detail.

5.2.1 Graph Attention Networks

The main component of GAT-Denoiser is a GAT. As mentioned in Section 4.3 *Graph Deep Learning*, GAT is an extension to GCN and adds attention (or weights) to neighbors for learning new node feature representations. Again, topology of the graph will not change but weighted averaging over the neighborhood will take place and this is what in denoising is a good idea.

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:

$$e_{ij} = a(Wh_i, Wh_j), \tag{5.3}$$

with a as the shared attentional mechanism $a : \mathbb{R}^{F'} \times \mathbb{R}^{F'} \mapsto \mathbb{R}$. But how to define the attentional mechanism? Veličković et al. [21] 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:

$$h'_i = \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij} W h_j\right), \tag{5.4}$$

with σ as some arbitrary activation function.

Multi-Head attention Motivated by Vaswani et al. [20], 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:

$$h'_i = \Big\Vert_{k=1}^{K} \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij}^k W^k h_j\right), \tag{5.5}$$

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: As in the last layer, dimension need to be corresponding to output dimension, concatenation is no longer plausible and averaging is used to match desired dimension.

5.2.2 Convolution

The second key component of GAT-Denoiser is convolution. Convolution is an important part in Signal Processing, if not the most important one. It allows to average an incoming signal, further, it is very popular in computer vision. Convolution commonly operators on pixel spaces, where every observation location or time slot gets one value assigned.

$$x \star k = y, \tag{5.6}$$

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. In Figure 5.2.2 an illustration of 1D convolution can be seen. In this example, kernel size is set to 3, n refers to input signal dimension and m to output signal dimension. In the example $n = m + 2$, so the convolved output signal will be decreased by 2. The concept of convolution can be extended to arbitrary dimensions.

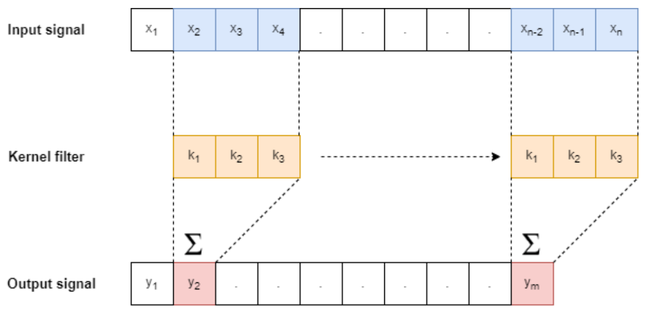


Figure 5.2: 1D Convolution

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

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 dimension will decrease.

5.2.3 U-Net

U-Net can boost performance of computed tomography reconstruction and is therefore our third component in GAT-Denoiser. It is a convolutional neural network, which is well suited for image segmentation in different domains. [15] 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 5.2.3.

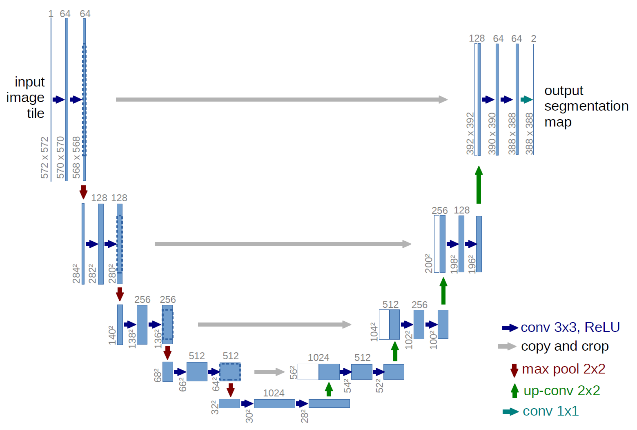


Figure 5.3: U-Net architecture [15, p 2, Fig. 1]
Number on top of boxes denotes channels, where numbers at bottom of boxes refer to input dimension.

During contracting path (left part) input dimension will be decreased and 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 feature map takes place, followed by a 2x2 convolution, which halves the number of channels. Then, concatenation with the corresponding feature map of contracting path is done (gray arrow in Figure 5.2.3), followed by again two 3x3 convolutions and ReLu. Final layer is a 1x1 convolution to map to desired output dimension and single output channel.

5.3 Architecture

Finally, all individual components are introduced and GAT-Denoiser architecture can be defined. First, different neural network layers will be explained and, second, loss and training. As defined in Equation 3.6, N is the number of observation and M observation dimension. Therefore, input (noisy sinogram) will be in $\mathbb{R}^{N \times M}$ as well as output (denoised sinogram).

5.3.1 Layers

In Figure 5.3.1, detailed GNN architecture can be seen. It is parametrized with *channels*, *heads* and *layers*. The number of channels in convolution can be increased with parameter *channels*. Further, *heads* determine the number of heads used in the GAT layers and parameter *layers* defines how many convolution and GAT layers are stacked together.

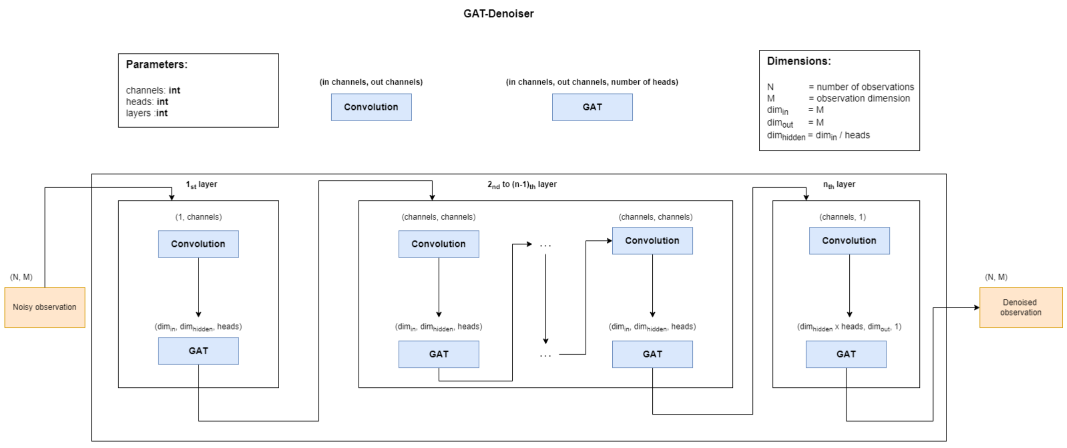

GAT-Denoiser
Parameters:
channels: int
heads: int
layers :int
(in channels, out channels)
Convolution
(in channels, out channels, number of heads)
GAT
Dimensions:
N = number of observations
M = observation dimension
dim$_{in}$ = M
dim$_{out}$ = M
dim$_{hidden}$ = dim$_{in}$ / heads
(N, M)
Noisy observation
1$_{st}$ layer
(1, channels)
Convolution
(dim$_{in}$, dim$_{hidden}$, heads)
GAT
2$_{nd}$ to (n-1)$_{th}$ layer
(channels, channels)
Convolution
(dim$_{in}$, dim$_{hidden}$, heads)
GAT
(channels, channels)
Convolution
(dim$_{in}$, dim$_{hidden}$, heads)
GAT
n$_{th}$ layer
(channels, 1)
Convolution
(dim$_{hidden}$ x heads, dim$_{out}$, 1)
GAT
(N, M)
Denoised observation

Figure 5.4: 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, dimension of convolved signal will not change. Convolution can be defined with different parameters, but output signal needs to have same dimension 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.

5.3.2 Training

As already mentioned, an end-to-end learning approach is used where quality of reconstruction is compared in the loss.
Therefore, the outcome of GAT-Denoiser is not directly part of the loss, but first reconstruction will be computed. Reconstructions can be nicely compared with the $\ell 2$-norm:

$$\mathcal{L} = \| x_i - Recon(GAT\text{-}Denoiser(A(x_i, \theta, s) + \eta)) \|_2^2 \tag{5.7}$$

As x_i is part of the loss, access to original object is needed during training.
Further, U-Net will be jointly used with FBP as reconstruction. For that to work, U-Net needs to be first pre-trained with the dataset and learned model can be plugged into GAT-Denoiser. One could also consider training U-Net model jointly, while training GAT-Denoiser.

6 Results

In this Chapter, results of GAT-Denoiser are presented. First, some small experiments are presented, which do not work on full dataset. These experiments have been done to find the best suitable architecture and to familiarize with the problem and its structure. Second, a few large experiments are presented, which work on full dataset. During small and large experiments, some interesting findings are illustrated and GAT-Denoiser is compared to U-Net and block-matching and 3D filtering (BM3D) [7].

6.1 Dataset

GAT-Denoiser is tested on LodoPaB-CT [13] dataset, which is a benchmark dataset for low-dose ct reconstruction methods and therefore well suited for our domain.
The dataset consists of 35820 train images and 3553 test images. All these images are having resolution 64.

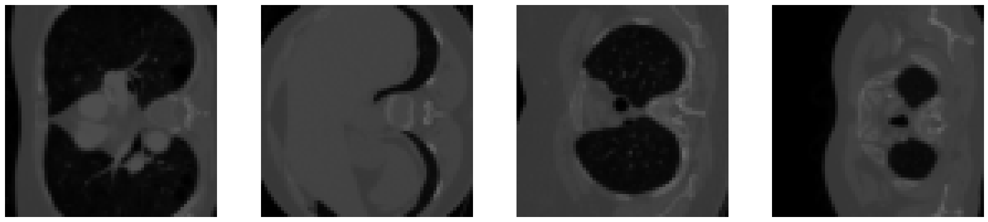


Figure 6.1: Some training images of LodoPaB-CT dataset.

6.2 Project setup

Python source code of the project is available on GitHub[3]. Several python packages are used in the code, Pytorch geometric[4] for neural network configuration, Operator Discretization

[3] https://github.com/cedricmendelin/master-thesis
[4] https://pytorch-geometric.readthedocs.io/en/latest/

Library[5] for Radon transform and FBP, and last, Weight & Bias[6] to gather results in a convenient way.
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 12 GB RAM have been used.

Radon transform: Radon transform will be applied in GAT-Denoiser and sampling points have been fixed to 64, so $s \in \mathbb{R}^64$. Further, projection angles θ are sampled within interval $[0, \pi]$. The number of observation is the size of our input graph, where typically 1024 nodes are used, but there are some experiments with other figures.
TODO: Mention batch size and batch learning

GAT settings: During experiments, in GAT-layers ELU was used as activation function. Further, dropout is used as regularization term and fixed to XX. During training, adam optimizer was with fixed learning rate 0.01 and weight decay 0.0005.

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.
To make notation clear, *SNR* in this Chapter refers to SNR computed from denoised reconstruction and original object and always shows an average on a test set for a specific algorithm or model. Contrary, SNR_y is used to express the level of noise during training or validation, where SNR is computed from clean and noisy observation.

U-Net training: As described in Section 5.2.3 *U-Net*, U-Net will be part of our model. Therefore, 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 $[0, -10]$ and was added to the observation (sinogram). In total, model was trained for 200 epochs.

BM3D: BM3D is a powerful and lightweight algorithm for image denoising. It is not neural network based and was the defacto state-of-the-art before neural network outperformed BM3D. Therefore, it is an interesting baseline to compare with.
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 sinogram* and *BM3d reconstruction* are used to distinguish between the two approaches.

[5] https://odlgroup.github.io/odl/
[6] https://wandb.ai/site

6.3 Small Experiments

For small 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.

6.3.1 Baseline

In Table 6.1, baseline results for FBP, BM3D sinogram, BM3D reconstruction and U-Net can be found. Noise was added to reach SNR_y of 0, -5, -10 and -15 dB.
As expected, reconstruction FBP performs the worst, as it only reconstructs from noisy observation. For higher SNR 0 dB BM3D outperforms U-Net, which lays down the power of BM3D. But, for lower SNR's -5 to -15 U-Net performs better and is the baseline to beat.

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	1041.6	4.81	1772.8	0.0	3105.6	-4.96	5495.2	-9.94
BM3D sinogram	662.2	9.30	909.5	6.13	1285.2	2.93	1884.9	-0.50
BM3D reconstruction	**604.2**	**10.15**	840.4	6.81	1283.1	2.89	2079.1	-1.42
U-Net	674.9	9.03	**711.5**	**8.17**	**711.5**	**8.17**	**1224.9**	**3.10**

Table 6.1: Baseline results small experiments: the best result in column is marked bold.

Further, for SNR$_y$0 dB, reconstruction of a single test image for all baseline algorithms is illustrated in Figure 6.3.1.

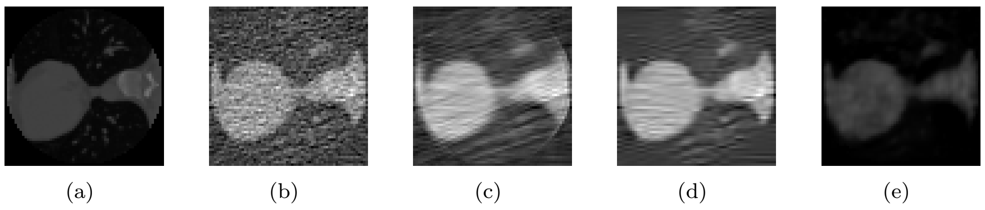


Figure 6.2: Example test image baseline reconstructions for SNR$_y$0 dB:
6.2(a) clean test image, 6.2(b) FBP reconstruction, 6.2(c) BM3D sinogram, 6.2(d) BM3D reconstruction, 6.2(e) U-Net reconstruction

6.3.2 Input graph structure

In the following Section, experiments with different input graphs are presented. As the domain of interest is the input graph and its size, and therefore the number of observations, 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. As described in Section 4.4.5 *Manifold of computed tomography and cryo-EM* the computed tomography underlying man-

ifold is assumed to be a circle in the clean case, therefore our graph is fixed as a circle graph. Moreover, learning is expected to fail when learning on a random graph.
Further, the neighborhood size for k-NN graphs is explored, as it is expected to be not easy to find good values for parameter k.

Does learning fail with random graph? Yes it does.
During this experiment, two models with different input graphs, but same size 1024 nodes 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 approximately connected to 1% of all available nodes. Consequently, both graphs have approximately the same amount of edges. In the experiment, random Erdős–Rényi graph fails to learn denoising and k-NN starts to capture some details. Table 6.2 shows Loss and SNR and in Figure 6.3.2 example reconstruction is illustrated.

Input graph	**Loss**	**SNR**
Erdős–Rényi graph with $p = 0.01$	1509.3	1.3
K-NN graph with $k = 10$	718.74	7.7

Table 6.2: Loss and SNR for random graph vs. k-NN graph.

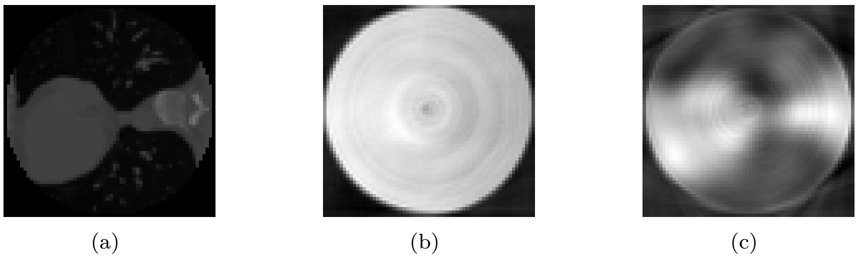


Figure 6.3: Example test image reconstructions for different input graphs:
Where 6.3(a) is clean object, 6.3(b) reconstruction with Erdős–Rényi input graph with $p = 0.01$ and 6.3(c) reconstruction K-NN input graph with $k = 10$.

Does result improve with large graph size? Slightly. For this question to answer, multiple models have been trained for graphs with size 512, 1024 and 2048 as well as different k-NN parameters $(2, 6, 8, 10, 20)$.
In Table 6.3 results are presented. So yes, larger graph size improve our model. But, training is also more expensive regarding execution time. Therefore, for all upcoming experiments graph size is fixed to 1024, as it looks like a good value in between, with good reconstruction result but also not too much training time.

Input graph size	**Loss**	**SNR**	**Training time (s)**
512	732.7	7.6	2678 s
1024	696.1	8.1	4216 s
2048	628.8	8.9	7609 s

Table 6.3: Loss and SNR for different input graph sizes, which refer to the average of different k-NN parameters.

What is a good parameter k for k-NN graph construction? TODO: Show result for k-nn 1-5 and argue why using 8 or 2

As described in Chapter 4.4.5 *Manifold of computed tomography and cryo-EM*, it is not easy to determine parameter k in general. Therefore, multiple models have been trained with different $k = (2, 6, 8, 10, 20)$ as well as different SNR_yrange (0 dB and -10 dB)

In Table 6.4, result of calculated models are illustrated. First, input graph resulting from $k = 2$ performs surprisingly good for SNR 0 dB as well as -10 dB.

K-NN parameter k	**SNR_y 0 dB**		**SNR_y -10 dB**	
	Loss	**SNR**	**Loss**	**SNR**
2	**609.2**	**9.15**	741.4	7.49
6	650.3	8.58	919.8	5.57
8	683.0	8.17	859.4	6.19
10	718.7	7.72	954.8	5.25
20	819.3	6.62	1028.4	4.63

Table 6.4: Different k-NN values for SNR_y0 dB and -10 dB

TODO: Argue why from now on k-nn = 8

6.3.3 GAT parameters

In this Section, experiments are executed with focus on GAT and it's parameters. Mainly, two parameters are important: number of heads and number of GAT layers.

To recap, the k-hop neighborhood in a GNN is defined by the number of layers. Therefore, number of layers is expected to keep rather low. The dataset has only resolution 64 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 divides 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.

During GAT experiments, graph size 1024 was used and parameter $k = 8$. Further, convolution was applied with a single channel, kernel size = 3 and padding 1. As focus is on GAT parameters, U-Net was not used during these experiments. 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 6.5.

Heads	**2 Layers**		**3 Layers**		**4 Layers**		**6 Layers**		**Average**	
	Loss	**SNR**	**Loss**	**SNR**	**Loss**	**SNR**	**Loss**	**SNR**	**Loss**	**SNR**
1	743.9	7.47	787.8	6.92	883.8	5.95	1527.3	1.18	**985.7**	**5.38**
2	697.1	8.00	764.1	7.22	883.0	5.99	1133.9	3.90	**869.5**	**6.28**
4	**646.7**	**8.66**	936.6	5.44	1071.1	4.28	1112.7	3.92	**941.8**	**5.58**
8	657.7	8.49	720.0	7.72	934.4	5.46	1666.7	0.86	**994.7**	**5.63**
16	650.2	8.59	650.2	6.35	884.2	5.92	1667.4	0.86	**1010.7**	**5.43**
Average	**679.1**	**8.24**	**809.9**	**6.73**	**931.3**	**5.52**	**1421.59**	**2.14**		

Table 6.5: Models trained with different GAT parameters: best and average result in column is marked bold.

The more layers the better result? Definitely not. As expected, when working with too many layers, denoising starts failing as it averages over a neighborhood, which is too large. In Table 6.5, 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. In the experiment, 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 7.47 dB where with 4 heads average SNR is 8.66.

How about some 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 as well. A small experiment with fixed network parameters and different dropout values $(0, 0.01, 0.03, 0.05, 0.1)$ have been calculated. 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.

6.3.4 Convolution parameters

TODO: Add Intro for conv. As convolution with multiple channels needs $layers > 2$, the best result for convolution
Multiple channels, different kernel-size and padding.
First results: 4 Layers, 2 Heads, k-n = 8.

Kernel and Padding	1 Channel		4 Channel		8 Channel		16 Channel	
	Loss	**SNR**	**Loss**	**SNR**	**Loss**	**SNR**	**Loss**	**SNR**
7 / 3	1212.6	3.28	1251.53	3.28	1266.7	2.89	1669.0	0.85
5 / 3	979.63	5.04	1588.67	0.99	1554.61	1.04	1667.84	0.86
3 / 1	823.29	6.57	956.34	5.27	1896.10	0.75	2300.93	0.03

Table 6.6: Baseline results large experiments: Best result in column is marked bold.

Second results, different graphs:

Channels	2 Layers		3 Layers		4 Layers	
	Loss	**SNR**	**Loss**	**SNR**	**Loss**	**SNR**
No Conv	768.4	4.58	1271.8	2.76	1031.3	7.15
1	647.33	8.63	786.6	6.94	823.3	0.0
4	762.82	7.23	798.6	6.80	956.3	0.8
8	689.42	8.08	786.9	7.24	1896.1	5.3
16	724.06	7.65	928.5	5.67	2300.9	6.6

Table 6.7: Baseline results large experiments: Best result in column is marked bold.

6.3.5 Loss

As introduced in Section 5.3.2 *Training*, during training loss is calculated based on quality of reconstruction, what is called end-to-end learning in this Thesis.
One could also think to compute loss from GAT-Denoiser output, on denoised observation level. For our LodoPaB-CT dataset, this would mean to compare clean sinogram with denoised sinogram:

$$\mathcal{L}_{sino} = \parallel p_i - \textit{GAT-Denoiser}(A(x_i, \theta, s) + \eta) \parallel_2^2 \tag{6.1}$$

Therefore, an experiment was set up to compare introduced loss introduced in Equation 5.7 with loss in Equation 6.1.

Is our end-to-end learning a good idea? Yes it is, but comes with a prize. For same network parameters and calculated for $\text{SNR}_y 0$, -5, -10 and -15 dB, experiments have been calculated once with $\mathcal{L}$ and once with $\mathcal{L}_{sino}$. Averaged results over SNR can be found in Table 6.8. 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 learning with $\mathcal{L}$, reconstruction of every sample needs to be done in every epoch, where training with $\mathcal{L}_{sino}$, reconstruction is not needed.
All in all, end-to-end learning with loss $\mathcal{L}$ shows great results and will be used in further experiments.

Loss training	**Average Loss**	**Average SNR**	**Training time (s)**
$\mathcal{L}$	762.6	7.34	1209 s
$\mathcal{L}_{sino}$	884.7	6.36	3812 s

Table 6.8: Average Loss and SNR of different training loss $\mathcal{L}$ and $\mathcal{L}_{sino}$

6.3.6 GAT-Denoiser components

TODO: show k-nn 2 and 8 (some in appendix) and argue that 2 is fine.
So far, no experiments with activated U-Net have been evaluated. The goal of previous experiments was to found suitable GAT-Denoiser neural network parameters. With these

last experiments for small LodoPaB-CT dataset, the different GAT-Denoiser components will be compared against each other. 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 overall settings of GAT-Denoiser has been defined with 2 layers, 4 heads, convolution with kernel size 3 and padding 1. K-NN parameter k was set to 8.
Further, the following models have been defined. First, *GAT* is the model, where no additional convolution and U-Net was used. Second, *GAT + Conv* refers to the model with GAT and convolution but no U-Net. Then, the two models have been combined with U-Net, resulting in two more models namely *GAT + U-Net* and the model with all components *Conv + GAT + U-Net*. The models, where U-Net is used and trained jointly with GAT-Denoiser is notated by *Unet**.
Overall, 6 models are evaluated and compared against baseline, results can be seen in Figure 6.3.6.

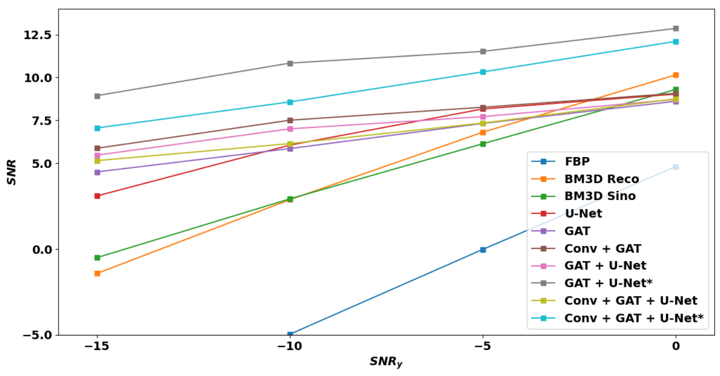


Figure 6.4: Resulting SNR of different models and baseline algorithms.

TODO: Add some visual results to Appendix.

6.4 Large Experiments

6.4.1 Baseline

In Table A.1, baseline results for large experiment can be found. As algorithms are fixed, it is expected that the result is pretty close to the one from the small experiment.
TODO: update with new numbers

Algorithm	**Input SNR 0 dB**		**Input SNR -5 dB**		**Input SNR -10 dB**		**Input SNR -15 dB**	
	Loss	**SNR**	**Loss**	**SNR**	**Loss**	**SNR**	**Loss**	**SNR**
FBP	1041.6	4.81	1772.8	0.0	3105.6	-4.96	5495.2	-9.94
BM3D sinogram	662.2	9.30	909.5	6.13	1285.2	2.93	1884.9	-0.50
BM3D reconstruction	**604.2**	**10.15**	840.4	6.81	1283.1	2.89	2079.1	-1.42
U-Net	674.9	9.03	**711.5**	**8.17**	**711.5**	**8.17**	**1224.9**	**3.10**

Table 6.9: Baseline results large experiments: Best result in column is marked bold.

TODO: Add some visual results to Appendix.

6.4.2 GAT-Denoiser components

TODO: Same same, but different. 20 epochs for all the models.

6.4.3 Last two models

TODO: Train for about 60 to 80 epochs and show best result. Select best 2 models from above.

7 Conclusion and Future Work

7.1 Conclusion

- GAT for denoising works with Manifold Assumption
- U-Net boosts performance
- U-Net could even be trained better (not SNR interval, longer)

7.2 Future Work

- Add Convolution in other way (in GAT directly? multiple Convolution layers before GAT?)
- Drop assumption about known angles.
- Move problem to 3D (first 3D CT, then cryo-EM)

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

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

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

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

[20] 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.

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

[22] 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.

[23] 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.

[24] 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 Appendix

A.1 Mathematical Tools

A.1.1 signal-to-noise-ratio (SNR)

A.1.2 Activation functions

A.1.2.1 ELU

A.1.2.2 ReLu

A.1.2.3 Leaky ReLu

A.1.2.4 Random Graph

A.1.3 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.1.4 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 converges. The algorithm will converge 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.

A.2 Small Experiment

A.2.1 GAT-Denoiser components

Algorithm	**SNR$_y$ 0 dB**	**SNR$_y$ -5 dB**	**SNR$_y$ -10 dB**	**SNR$_y$ -15 dB**
	SNR	**SNR**	**SNR**	**SNR**
FBP	4.80	-0.017	-4.96	-9.94
U-Net	9.03	8.17	6.06	3.10
BM3D Reco	10.15	6.81	2.88	-1.42
BM3D Sino	9.30	6.14	2.93	-0.50
GAT	8.61	7.32	5.86	4.50
Conv + GAT	9.08	8.27	7.51	5.88
Gat + Unet	8.72	7.72	7.01	5.48
Gat + Unet*	12.86	11.52	10.84	8.94
Conv + Gat + Unet	8.75	7.34	6.14	5.15
Conv + Gat + Unet*	12.10	10.33	8.58	7.06

Table A.1: Small Experiment: overall GAT-Denoiser components vs. Baseline

A.3 Large Experiment: additional results

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 .