Graph Denoising Networks: A Deep Learning Framework for Equity Portfolio Construction

Edward Turner
Oxford-Man Institute of Quantitative Finance
University of Oxford
Oxford, United Kingdom
edward.turner01@outlook.com

Mihai Cucuringu
Department of Statistics & Mathematical Institute
Oxford-Man Institute of Quantitative Finance
University of Oxford
Oxford, United Kingdom
mihai.cucuringu@stats.ox.ac.uk

ABSTRACT

Graph-based deep learning is a rapidly evolving and practical field due to the ubiquity of graph data and its flexible topology. Although many graph learning frameworks show impressive capabilities, their outputs begin to deteriorate for sufficiently noisy data. In this paper, we look to overcome this shortcoming by introducing the *Graph Denoising Network*, which combines denoising diffusion methods with graph models in a compounding manner. We prove under certain conditions that this can be construed as an MCMC approach to learning and sampling from the true data distribution. When testing on a graph built from financial returns, we obtain Sharpe Ratios of up to 4.4, and consistently above 2. Compared to a baseline graph convolutional network, we find noticeable improvement and statistical evidence to conclude that graph denoising networks improve performance and attain significant economic benefits. Our findings are applicable to other domains that employ noisy graph-based data, potentially in a time-dependent context.

CCS CONCEPTS

• **Mathematics of computing** → **Markov-chain Monte Carlo methods**; • **Computing methodologies** → *Neural networks*; *Learning latent representations*; • **Applied computing** → Economics.

KEYWORDS

Graph neural networks, Denoising diffusion, Markov-chain Monte Carlo, Time-series, Financial returns

ACM Reference Format:
Edward Turner and Mihai Cucuringu. 2023. Graph Denoising Networks: A Deep Learning Framework for Equity Portfolio Construction. In *4th ACM International Conference on AI in Finance (ICAIF '23), November 27–29, 2023, Brooklyn, NY, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3604237.3626903

1 INTRODUCTION

Graph representation learning (GRL) is a modern, rapidly developing field with the aim of projecting a feature rich graph (e.g. a transport network with vertices representing cities, possibly holding information such as size or population, and edges representing connection via land or air travel) onto a lower-dimensional manifold where the data lies; the latent space. A key requirement is that the mapping be invariant under graph isomorphism (i.e. relabelling the vertices has no impact). This permutation invariance is one of the features that sets GRL apart from other machine learning domains. The latent space aims to provide a better environment for downstream inference tasks, typically involving vertex prediction (predicting labels for some or all of the vertices) or relation prediction (predicting edge existence between vertex pairs) [6]. In this paper we focus on vertex prediction tasks.

A seemingly unrelated area, but one we look to bring to GRL, are generative models, specifically denoising diffusion models. Denoising diffusion models have emerged as a powerful new family of deep generative models with high performance in many applications, including image denoising and synthesis [28]. They have broken the long-time dominance of generative adversarial networks (GANs) and are now industry leading [28]. Popular models such as DALL-E 2 from OpenAI and Imagen from Google rely on denoising diffusion, producing state-of-the-art synthesis results [17, 20].

We hypothesise that the correct utilisation of denoising diffusion models in GRL will allow for cleaner data to be passed into GRL models, resulting in latent representations closer to the true underlying data, thus increasing performance. To test this hypothesis, we consider a detailed application to stock market data. This domain inspired the idea of bringing denoising diffusion models to GRL due to their capabilities of removing Gaussian noise. An issue with financial data is the high noise-to-signal ratio. This noise is due to inherently highly-volatile latent market factors and a complex market outlook with future price returns influenced by many variables, ranging from news and current affairs to weather and macroeconomic variables. Typically, the noise destroys any kind of representation learning, hence the reason why linear models are so commonly relied upon in industry in real-time production environments. If one passes raw data as input, a typical deep learning model will learn and exploit a set of spurious correlations, rather than any useful time-persistent alpha generating representation. With this in mind, we view financial data as a highly relevant application for graph denoising networks (GDNs). It allows one to test if the model removes enough noise that a meaningful representation is ultimately learned, thus enabling the applicability of said model to suitably defined downstream tasks of interest.

Summary of main contributions. We introduce a novel graph denoising network, and propose a novel training method for it,

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
ICAIF '23, November 27–29, 2023, Brooklyn, NY, USA
© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-0240-2/23/11…$15.00
https://doi.org/10.1145/3604237.3626903

along with code for its implementation. The model takes a semi-supervised approach, combining two areas that, to the best of our knowledge, have not been previously connected. It brings the unsupervised generative methods of denoising diffusion to vertex feature creation, in an attempt to learn the original generative process. Our combination of these two areas is via *compounded message passing*, rather than just separate model blocks passed sequentially. In Theorem 3, we prove under the right assumptions that this can be viewed as a Markov-chain Monte Carlo (MCMC) approach to finding and then generating data from the true underlying distribution. To this end, our model can be seen as its own class of graph-based models, rather than simply an adaptation of spatio-temporal methods. We demonstrate the efficacy of our approach on a financial data set of equity returns, where it attains significantly higher economic benefits and Sharpe Ratios, when compared to a baseline graph convolutional network. Our code and data are fully available at https://github.com/edwardbturner/GDN_finance.

2 PRELIMINARIES

2.1 Markov chains

Here we introduce (time-homogeneous) Markov chains with possibly infinite state spaces to later use in Section 3. Let $(X_t)_{t\geq 1}$ be a sequence of random variables on the probability space $(\Omega, \mathcal{B}_\Omega, \mu)$ with Ω possibly infinite, $\mathcal{B}_\Omega$ the Borel σ-algebra on Ω and μ a probability measure. Further, let $S_\mu = supp(\mu) := \{B \in \mathcal{B}_\Omega \,|\, \mu(B) > 0\}$. We say $(X_t)_{t\geq 1}$ is a Markov chain with memory m if:

$$K(X_{1:t-1}, A) := \mathbb{P}(X_t \in A \,|\, X_{t-1}, ..., X_1) \tag{1}$$

$$= P(X_t \in A \,|\, X_{t-1}, ..., X_{t-m}) \;\; \forall A \in \mathcal{B}_\Omega\,, \tag{2}$$

where $K(X_{1:t-1}, A)$ is the transition kernel. In the case of $m = 1$ $(X_t)_{t\geq 1}$ is called a Markov chain and we abbreviate $K(X_{1:t-1}, A)$ with $K(X_{t-1}, A)$.

A Markov chain is ergodic if it is positive Harris recurrent:

$$\mathbb{P}(X_t \in A \text{ infinitely often} \,| X_1 = x) = 1 \;\; \forall x \in \Omega\,, \;\; \forall A \in S_\mu\,, \tag{3}$$

and aperodic: there are no sections of Ω that it can only visit at certain times.

The distribution μ is invariant for the Markov chain $(X_t)_{t\geq 1}$ if:

$$\mu(A) = \int_\Omega K(x, A)\mu(dx) \;\; \forall A \in \mathcal{B}_\Omega\,, \tag{4}$$

and is an equilibrium distribution if for μ-almost all $x \in \Omega$:

$$\lim_{t\to\infty} \mathbb{P}(X_t \in A \,|\, X_1 = x) = \mu(A) \;\; \forall A \in \mathcal{B}_\Omega\,. \tag{5}$$

Finally, a Markov chain $(X_t)_{t\geq 1}$ with invariant distribution μ is strongly irreducible if:

$$K(x, A) > 0 \;\; \forall x \in \Omega\,, \;\; \forall A \in S_\mu\,. \tag{6}$$

2.2 Graph neural networks

We consider graphs of the form $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W}, \mathcal{X})$ where $\mathcal{V}$ is the vertex set, $\mathcal{E}$ is the edge set, $\mathcal{W}$ is the (possibly weighted) adjacency matrix and $\mathcal{X}$ is a vertex feature matrix. Graph neural networks (GNNs) are a deep learning approach to GRL that work via a multi-layer message passing framework, comprising of an initial layer and L further layers [6]. For an input graph $\mathcal{G}$, the kth layer contains n nodes and may be represented as a matrix, $\mathcal{H}^{(k)} = \left(h_1^{(k)\,T}, ..., h_n^{(k)\,T}\right)^T \in \mathbb{R}^{n\times d_k}$, with d_k being the dimension of the graph embedding in the kth layer. Here $h_i^{(k)} \in \mathbb{R}^{1\times d_k}$ is a row vector that represents the ith node in the kth layer which corresponds to the ith vertex of $\mathcal{G}$. A general GNN is initialised by the following iteration:

$$\mathcal{H}^{(0)} = \mathcal{X}\,, \tag{7}$$

$$h_i^{(k+1)} = \phi\Big(h_i^{(k)}, \bigoplus_{j\in\mathcal{N}(i)} \psi\big(h_j^{(k)}\big)\Big)\,, \tag{8}$$

where $\mathcal{N}(i) := \{j : e_{i,j} \in \mathcal{E}\}$ is the neighbour set of the ith vertex, $\oplus$ is a differentiable aggregation function that may depend on k, ψ is a differentiable transformation that may depend on k or $\mathcal{W}$ and ϕ is a differentiable update function that may also depend on k or $\mathcal{W}$. The first layer is set to be the vertex features and then for each following layer a vertex's embedding is based on its previous embedding updated with an aggregation of its neighbouring vertices' transformed previous embeddings. We take $\mathcal{H}^{(L)}$ to be our final low-dimension embedding of the graph, the latent representation.

To avoid overfitting a self-loop approach is often used, here Eqn (8) becomes:

$$h_i^{(k+1)} = \bigoplus_{j\in\mathcal{N}(i)\cup\{i\}} \psi\big(h_j^{(k)}\big)\,, \tag{9}$$

that is, the update and aggregate functions are replaced by a single aggregate function [6]. This results in a vertex's previous embedding being viewed with equal importance (modulo any ψ dependence on $\mathcal{W}$) as its neighbours.

2.3 Graph convolutional networks

Graph convolutional networks (GCNs) were one of the first models that brought GRL to the forefront of machine learning. The methodology GCNs rely on was introduced by Bruna et al. and then extended by Defferrard et al. [2, 4]. The framework takes inspiration for its aggregate functions from Fourier transforms in the spectral domain [29]. Defferrard et al. define the normalised graph Laplacian, $\mathcal{L} := I_n - \mathcal{D}^{-\frac{1}{2}}\mathcal{W}\mathcal{D}^{-\frac{1}{2}}$ where $\mathcal{D}_{ii} = \Sigma_j \mathcal{W}_{ij}$, and then consider its spectral decomposition $\mathcal{L} = U\Lambda U^T$, with $\Lambda \in \mathbb{R}^{n\times n}$ the diagonal matrix of eigenvalues of $\mathcal{L}$ and $U \in \mathbb{R}^{n\times n}$ the matrix of eigenvectors of $\mathcal{L}$, known as the graph Fourier basis [4, 22]. For $x \in \mathbb{R}^n$, a single feature vector, they use a self-loop approach with a filter matrix, $g_\theta := \text{diag}(\theta) \in \mathbb{R}^{n\times n}$, $\theta \in \mathbb{R}^n$, for the aggregate function, defined by:

$$g_\theta * x := U g_\theta U^T x\,. \tag{10}$$

Thus g_θ acts in the eigenspace of $\mathcal{L}$ and hence we may view it as a function of its eigenvalues: $g_\theta(\Lambda)$. The eigendecomposition of $\mathcal{L}$ and subsequent evaluation of Eqn (10) is computationally expensive, thus similarly to classical CNNs work was done to to localise the filter and reduce the number of parameters, subsequently reducing the computational complexity. Hammond et al. proposed truncating the Chebyshev polynomial expansion of $g_\theta(\Lambda)$ up to the Kth order [7], which yields:

$$g_\theta(\Lambda) \approx \sum_{k=0}^{K} \theta'_k T_k(\tilde{\Lambda})\,, \tag{11}$$

where θ'_k is a vector of trainable Chebyshev coefficients and $T_k(\tilde{\Lambda}) \in \mathbb{R}^{n\times n}$ is the Chebyshev polynomial of order k evaluated at $\tilde{\Lambda} := \frac{2}{\lambda_{\max}}\Lambda - I_n$ with $\lambda_{\max}$ denoting the largest eigenvalue of $\mathcal{L}$ [8, 29]. The Chebyshev polynomials are recursively defined by $T_0(x) = 1$, $T_1(x) = x$ and $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)\ \forall k \geq 2$ [7]. Note, Eqn (11) is now K-localised since it is a K^{th}-order polynomial in the Laplacian, that is it only depends on vertices that are at maximum K steps away from the central vertex.

Defferrard et al. combine Eqn (10) with Eqn (11) to obtain their GCN message passing framework:

$$g_\theta * x \approx U\left(\sum_{k=0}^{K}\theta'_k T_k(\tilde{\Lambda})\right)U^T x \tag{12}$$

$$= \sum_{k=0}^{K}\theta'_k T_k(\tilde{\mathcal{L}})x\,, \tag{13}$$

where line (13) follows from noting $(U\Lambda U^T)^k = U\Lambda^k U^T$ and defining $\tilde{\mathcal{L}} := \frac{2}{\lambda_{\max}}\mathcal{L} - I_n$ [4]. Kipf et al. introduced a simplified version of Eqn (13) by restricting to $K = 1$ and approximating $\lambda_{\max} \approx 2$, resulting in:

$$g_\theta * x \approx \theta'_0 x + \theta'_1(\mathcal{L} - I_n)x \tag{14}$$

$$\approx \theta(I_n + \mathcal{D}^{-\frac{1}{2}}\mathcal{W}\mathcal{D}^{-\frac{1}{2}})x\,, \tag{15}$$

where line (15) follows by setting $\theta = \theta'_0 = -\theta'_1$ [12]. Normalising $I_n + \mathcal{D}^{-\frac{1}{2}}\mathcal{W}\mathcal{D}^{-\frac{1}{2}}$ and generalising Eqn (15) to handle a vertex feature matrix results in their GCN:

$$h_i^{(k+1)} = \sigma\left(\sum_{j\in\mathcal{N}(i)\cup\{i\}} c_{i,j}h_j^{(k)}\Theta^{(k)}\right), \tag{16}$$

which can be succinctly written to update the whole layer at once:

$$\mathcal{H}^{(k+1)} = \sigma\left(\widetilde{\mathcal{D}}^{-\frac{1}{2}}\widetilde{\mathcal{W}}\widetilde{\mathcal{D}}^{-\frac{1}{2}}\mathcal{H}^{(k)}\Theta^{(k)}\right), \tag{17}$$

where $\widetilde{\mathcal{D}}^{-\frac{1}{2}}\widetilde{\mathcal{W}}\widetilde{\mathcal{D}}^{-\frac{1}{2}}$ is the symmetric normalisation of the weight matrix with added self-loops, specifically $\widetilde{\mathcal{W}} = \mathcal{W} + I_n$ and $\widetilde{\mathcal{D}}_{ii} = \Sigma_j\widetilde{\mathcal{W}}_{ij}$ [12]. Here $c_{i,j}$ is a normalisation constant determined by $\widetilde{\mathcal{D}}^{-\frac{1}{2}}\widetilde{\mathcal{W}}\widetilde{\mathcal{D}}^{-\frac{1}{2}}$, $\Theta^{(k)} \in \mathbb{R}^{d_k\times d_{k+1}}$ is a trainable weight matrix and σ is an element-wise non-linear activation function, taken to be ReLU by Kipf et al. [12]. This model is able to efficiently encode both the local graph structure and the vertex features [12, 13]. Applying K layers of Eqn (17) achieves a similar effect as one passing of the K-order convolution, while the layer-wise linear structure is parameter-economic and highly efficient for large-scale graphs since the order of the approximation is limited to one [29].

2.4 Denoising diffusion probabilistic models

Diffusion models are latent variable models where for input data, $x_0 \in \mathbb{R}^n$, latents are created, $x_1, ..., x_T \in \mathbb{R}^n$, and the model is $p_\theta(x_0) := \int p_\theta(x_{0:T})\,dx_{1:T}$ [24]. Denoising diffusion models are a family of probabilistic generative models that progressively destroy input data by injecting noise [28]. They then learn to reverse this process for sample synthesis or other applications such as sample denoising. They are the current cutting edge for generative models and have been shown as far superior to GANs. Their benefit as a generative model is attributed to avoiding mode collapse and more stable training [15].

Ho et al. introduced denoising diffusion probabilistic models (DDPMs) where the original application was data generation [9]. A DDPM makes use of two Markov chains: a forward chain that perturbs data to noise, and a reverse chain that converts noise back to data [28]. The former is typically a multi-step addition of Gaussian noise, while the latter reverses the former by learning transition kernels parameterised by a deep neural network [28].

DDPMs are distinguished from other latent variable models due to the approximate posterior $q(x_{1:T}|x_0)$, called the forward (diffusion) process, defined as a Markov chain that gradually adds Gaussian noise to the data according to a variance schedule $\beta_1, ..., \beta_T \in (0, 1)$ [9]. For input data $x_0 \in \mathbb{R}^n$, the latents $x_1, ..., x_T \in \mathbb{R}^n$ have the following distributions:

$$q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1})\,, \tag{18}$$

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t;\ \sqrt{1-\beta_t}\,x_{t-1},\ \beta_t I_n)\,. \tag{19}$$

Since $q(x_{t-1}|x_t) = q(x_t|x_{t-1})q(x_{t-1})/q(x_t)$ it is intractable to calculate $q(x_{t-1}|x_t)$ despite having $q(x_t|x_{t-1})$, as one would also need to know $q(x_{t-1})$ and $q(x_t)$, which due to marginalisation both depend on $q(x_0)$, the very distribution we are trying to estimate. To overcome this we approximate $q(x_{t-1}|x_t)$ with a reverse process instead. The joint distribution $p_\theta(x_{0:T})$ is called the *reverse process*, and it is defined as a Markov chain with learned Gaussian transitions starting with $p(x_T) = \mathcal{N}(x_T;\ 0,\ I_d)$. Then we have:

$$p_\theta(x_{0:T}) = p(x_T)\prod_{t=0}^{T-1} p_\theta(x_t|x_{t+1})\,, \tag{20}$$

$$p_\theta(x_t|x_{t+1}) = \mathcal{N}(x_t;\ \mu_\theta(x_{t+1}, t+1),\ \sigma^2_{t+1}I_n)\,, \tag{21}$$

where $\sigma_1^2, ..., \sigma_T^2$ are time dependent constants and $\mu_\theta(x_{t+1}, t+1)$ is estimated by a deep neural network [9]. Typically the U-Net architecture is used, we cover the details of this in Section 3 [18].

The loss Ho et al. aim to minimise is the variational lower bound:

$$L_{\text{vlb}}(\theta) := \mathbb{E}_q\left[-\log\frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)}\right] \tag{22}$$

$$= \mathbb{E}_q\Big[-\log\left(p_\theta(x_0|x_1)\right) + D_{KL}(q(x_T|x_0)||p_\theta(x_T)) + \sum_{t=1}^{T-1} D_{KL}(q(x_t|x_{t+1}, x_0)||p_\theta(x_t|x_{t+1}))\Big]\,, \tag{23}$$

where D_{KL} is the Kullback–Leibler divergence. Defining $\alpha_t := 1-\beta_t$ and $\bar{\alpha}_t := \prod_{s=1}^{t}\alpha_s$, they approximate $L_{\text{vlb}}(\theta)$ with:

$$L_{\text{simple}}(\theta) := \mathbb{E}_{t,x_0,\epsilon}\left[\left\|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t)\right\|_2^2\right], \tag{24}$$

where the following reparameterisation has been used:

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t)\right). \tag{25}$$

This allows the mean squared error (MSE) of $\epsilon_\theta(x_t, t)$ to be targeted rather than the MSE of $\mu_\theta(x_t, t)$ which Ho et al. find to improve results [9]. The derivations of lines (23) and (25) are shown in the original DDPM paper [9].

2.5 Denoising diffusion restoration models

Firstly we introduce the idea of linear inverse problems and then discuss their relevance to denoising. A general linear inverse problem takes the form:

$$y = Hx_0 + z\,, \tag{26}$$

where $y \in \mathbb{R}^m$ is the noisy observation of the true signal $x_0 \in \mathbb{R}^n$, $H \in \mathbb{R}^{m\times n}$ is a known linear degradation matrix and $z \sim \mathcal{N}(0, \sigma_y^2 I_m)$ is a Gaussian noise term with known variance [11]. Given we observe y we aim to sample from the posterior $q(x_0|y) = q(y|x_0)q(x_0)/q(y)$ to reconstruct x_0. As with DDPMs, in general we do not know $q(x_0)$ and thus look to sample from an approximate posterior, $p_\theta(x_0|y)$, instead.

Denoising diffusion restoration models (DDRMs) were introduced by Kawar et al. [11], they use an unsupervised approach to solving linear inverse problems in an attempt to denoise a sample to the desired output, conditioned on the measurements and the inverse problem [11]. DDRMs take a Bayesian approach to solving general linear inverse problems by using a DDPM, pre-trained on the same domain, as a prior [10]. They have been shown to have impressive practical results, such as speech enhancement and strong-lense source recognition in Physics [10, 21].

Kawar et al. introduce computationally efficient results in the singular value space of H however we only require the denoising case: $H = I_n$, thus we make no further comments on the singular value decomposition specifics. The DDRM framework is the same as a DDPM but with conditional processes:

$$q(x_{1:T}|x_0, y) = q(x_T|x_0, y) \prod_{t=0}^{T-1} q(x_t|x_{t+1}, x_0, y)\,, \tag{27}$$

$$p_\theta(x_{0:T}|y) = p_\theta(x_T|y) \prod_{t=0}^{T-1} p_\theta(x_t|x_{t+1}, y)\,. \tag{28}$$

Note in Eqn (27) the backward conditionals, $q(x_t|x_{t+1}, x_0, y)$, are used instead of the forward conditionals, $q(x_{t+1}|x_t, x_0, y)$. However, thanks to Bayes rule, which is now feasible due to $q(x_t|x_0, y)$ and $q(x_{t+1}|x_0, y)$ being tractable, the forward conditionals are also well defined. Kewar et al. then define the forward process for $H = I_n$ by:

$$q(x_T|y) = \mathcal{N}(x_T;\, y,\, \sigma_T^2 - \sigma_y^2)\,, \tag{29}$$

and if $\sigma_t \geq \sigma_y$ then:

$$q(x_t|x_{t+1}, x_0, y) = \mathcal{N}(x_t;\ (1-\eta_b)x_0 + \eta_b y,\ \sigma_t^2 - \sigma_y^2\eta_b^2)\,, \tag{30}$$

otherwise for $\sigma_t < \sigma_y$:

$$q(x_t|x_{t+1}, x_0, y) = \mathcal{N}\Big(x_t;\ x_0 + \sqrt{1-\eta^2}\sigma_t \frac{y - x_0}{\sigma_y},\ \eta^2\sigma_t^2\Big)\,, \tag{31}$$

where σ_t corresponds to $\tilde{\alpha}_t$ in DDPMs, thus it represents the noise at step t. $\eta \in (0, 1]$ sets the transition variance with η and η_b both possibly depending on σ_t and σ_y [11].[1]

For $H = I_n$ the DDRM reverse process is defined as in Eqn (29) to Eqn (31) but swapping x_0 with $\tilde{x}_{\theta,t}$, where $\tilde{x}_{\theta,t} := f_\theta(x_{t+1}, t+1)$

[1] Here we assume $\sigma_T \geq \sigma_y$, Kawar et al. argue this is a valid assumption as σ_T can be set sufficiently large [11].

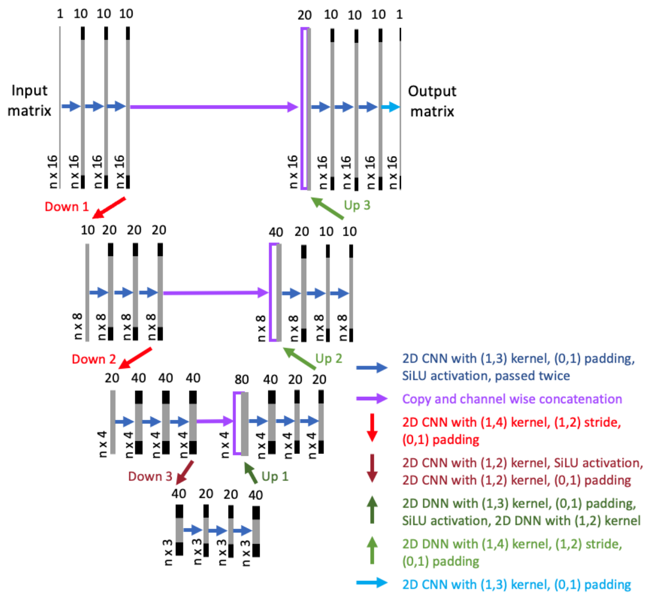

1 10 10 10
20 10 10 10 1
Input matrix
Output matrix
n x 16
Down 1
Up 3
10 20 20 20
40 20 10 10
n x 8
Down 2
Up 2
20 40 40 40
80 40 20 20
n x 4
Down 3
Up 1
40 20 20 40
n x 3
2D CNN with (1,3) kernel, (0,1) padding, SiLU activation, passed twice
Copy and channel wise concatenation
2D CNN with (1,4) kernel, (1,2) stride, (0,1) padding
2D CNN with (1,2) kernel, SiLU activation, 2D CNN with (1,2) kernel, (0,1) padding
2D DNN with (1,3) kernel, (0,1) padding, SiLU activation, 2D DNN with (1,2) kernel
2D DNN with (1,4) kernel, (1,2) stride, (0,1) padding
2D CNN with (1,3) kernel, (0,1) padding

Figure 1: Our PIU-Net architecture for a vertex feature matrix of n vertices and feature size $d = 16$. Each vertical bar represents the convolved matrix at that point with black ends representing padding. The number above each bar is the number of channels at that point. (Figure is an adaptation of one in the original U-Net paper [18].)

for some deep neural network $f_\theta(\cdot,\cdot)$ with a trainable parameter θ [11]. Kawar et al. then use the conditional equivalent of Eqn (23):

$$L_{\text{cvlb}}(\theta) := \mathbb{E}_q\Big[-\log\left(p_\theta(x_0|x_1, y)\right) + D_{KL}(q(x_T|x_0, y)||p_\theta(x_T|y)) + \sum_{t=1}^{T-1} D_{KL}(q(x_t|x_{t+1}, x_0, y)||p_\theta(x_t|x_{t+1}, y))\Big]\,. \tag{32}$$

Theorem 1. *If $f_\theta(\cdot, t)$ and $f_\theta(\cdot, t')$ do not share weights for $t \neq t'$ then for $\eta = 1$ and $\eta_b = \frac{2\sigma_t^2}{\sigma_t^2+\sigma_y^2}$ the DDRM loss, $L_{\text{cvlb}}(\theta)$, is equivalent to the DDPM loss, $L_{\text{vlb}}(\theta)$.*

Proof. In Appendix C of the original DDRM paper [11]. □

Theorem 1 is what gives DDRMs their practical usefulness. It allows the function $\mu_\theta(x_t, t)$ from a pre-trained DDPM to be used for the DDRM $f_\theta(x_t, t)$ function. Thus we can now sample from $p_\theta(x_0|y)$ without having to re-train the DDRM for each observed y. Equally all the hyper-parameters and degradation type can be changed without the need for retraining. In order for the $f_\theta(x_t, t) = \mu_\theta(x_t, t)$ assumption to be valid, one needs to train the DDPM on the same domain as that of the observed data.

3 GRAPH DENOISING NETWORKS

The first component of our model is a permutation invariant deep neural network that serves as the DDPM $\mu_\theta(x_t, t)$ function and thus from Theorem 1 also the DDRM $f_\theta(x_t, t)$ function. We take inspiration from the original DDPM paper where a U-Net is used; this is a deep neural network originally introduced for medical imaging [9, 18]. U-Net works by convolving the input down in stages to a bottleneck point, and then upscaling by concatenating each downscaled step with an upsample [18].

We introduce Permutation Invariant U-Net (PIU-Net), which, to the best of our knowledge, is a novel architecture. Having seen the success of Yu et al.'s ST-GCN, where they create permutation invariant CNNs by adapting to a 1-D kernel, we create PIU-Net with a similar ideology [29]. Namely we adapt all the U-Net CNNs to have 1-D kernels, strides and padding. At the same time, we adapt the model to handle non-square inputs, the only requirement is $W \equiv 0 \pmod 8$, where W is the width of the input matrix.

Figure 1 shows the full model architecture, here CNNs with a stride of (1, 2) are used for our downsampling and deconvolutional neural networks (DNNs), as introduced by Zeiler et al., are used for our upsampling [30]. As in the original DDPM paper we use a sinusoidal embedding with a single hidden layer multi-layer perceptron (MLP) to condition the backwards process on the time step [9]. The resulting tensors are added channel-wise to the PIU-Net output at each level.

Now that we have a permutation invariant model to train a DDPM, we define our full GDN. Figure 2 shows an overview. The pipeline starts by taking the vertex features on day t, $\mathcal{X}^{(t)}$, and using them as the noisy observation, $y^{(t)}$, in the DDRM. To sample $X_0^{(t)} \sim p_\theta(x_0|y^{(t)})$, a denoised version of $y^{(t)}$ as in Eqn (26), we require a trained $\mu_\theta(\cdot,\cdot)$ from the DDPM to use as the DDRMs $f_\theta(\cdot,\cdot)$. This, however, is an issue since training $\mu_\theta(\cdot,\cdot)$ requires samples from the true underlying distribution, $x_0^{(t)} \sim q(x_0)$. To overcome this, we use past outputs of our GCN as an approximation. Letting $\hat{x}_0^{(t)}$ denote the GCN output from day t we use $\hat{x}_0^{(t-1)}, ..., \hat{x}_0^{(t-lb)}$ as approximate samples from $q(x_0)$, where lb is a lookback training parameter.[2] This allows the DDPM to learn $\tilde{\mu}_\theta(\cdot,\cdot) \approx \mu_\theta(\cdot,\cdot)$. We can then use the DDRM with $\tilde{\mu}_\theta(\cdot,\cdot)$ to sample $\tilde{X}_0^{(t)} \sim \tilde{p}_\theta(x_0|y^{(t)})$, an approximation of $p_\theta(x_0|y^{(t)})$, giving an approximately denoised version of $\mathcal{X}^{(t)} = y^{(t)}$.

The intuition behind this is that the GCN guides the direction of the true underlying distribution for the DDPM to train on. This follows as GRL models typically assume the latent representation is more informative of the true distribution than the vertex features [6]. The key to GDNs is then the cyclical compounding framework, the DDRM component removes noise so as training progresses the GCN outputs become progressively closer to the true denoised distribution, allowing future DDRM samples to be closer to the true denoised data. Formally the GDN makes two assumptions:

Assumption 1. *$\mathcal{X} = \mathrm{GCN}(x_0) + Z$ where $Z = (z_{i,j}) \in \mathbb{R}^{n\times d}$ with $z_{i,j} \sim \mathcal{N}(0, \sigma_y^2)$.*

This stems from using $\mathcal{X} = y^{(t)}$ and our linear degradation assumption in Eqn (26). Data from many domains should satisfy Assumption 1; here, we focus on applying GDNs to financial returns data where we consider $\mathcal{X}$ to be log-returns. Thus Assumption 1 translates to saying that each stock's return k days ago, $\mathcal{X}_{s,k}$, can be viewed as an aggregation of signals from its neighbourhood, $(\mathrm{GCN}(x_0))_{s,k}$, for tomorrow's return plus some noise, $z_{s,k}$. This is

[2]Here we train on the most recent lb days to avoid concept drift. To see full training details and reproduce our results refer to the provided code.

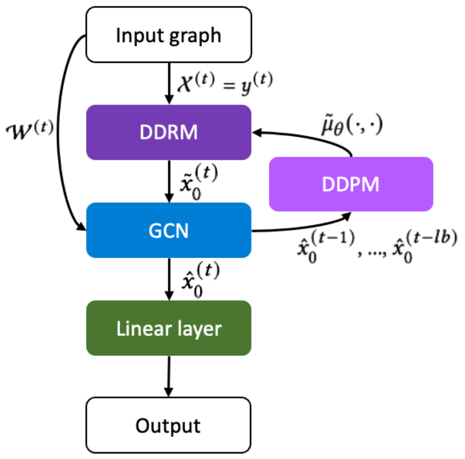

Input graph
$\mathcal{X}^{(t)} = y^{(t)}$
$\mathcal{W}^{(t)}$
DDRM
$\tilde{\mu}_\theta(\cdot,\cdot)$
$\tilde{x}_0^{(t)}$
DDPM
GCN
$\hat{x}_0^{(t-1)}, ..., \hat{x}_0^{(t-lb)}$
$\hat{x}_0^{(t)}$
Linear layer
Output

Figure 2: The GDN framework where superscripts refer to the day the data comes from. Here $\hat{x}_0^{(t-1)}, ..., \hat{x}_0^{(t-lb)}$ are previous GCN outputs used for training the DDPM, which then passes the trained $\tilde{\mu}_\theta(\cdot,\cdot)$ to the DDRM. This in turn allows us to sample from $\tilde{p}_\theta(x_0|y^{(t)})$, obtaining $\tilde{x}_0^{(t)}$, which we hypothesise will be a denoised version of $\mathcal{X}^{(t)} = y^{(t)}$.

a standard way to view returns with Campbell et al.'s *Econometrics of Financial Markets* textbook arguing returns form of a signal, representing the expected return based on fundamental factors such as earnings and dividends, plus a noise, from the unpredictable variation in returns due to factors such as news and sentiment [3].

Assumption 2. $\hat{X}_0^{(t-k)} \overset{d}{\approx} q(x_0) \quad \forall k \in \{1, ..., lb\}$.

The intuition behind Assumption 2 being valid, and why we propose the compounding model, comes from viewing $\hat{x}_0^{(t)}$ (the observed $\hat{X}_0^{(t)}$ values) as samples from a Markov chain with memory lb. That is consider $(\hat{X}_0^{(t)})_{t\geq1}$, the random sequence of DDRM outputs after passing the GCN module. The stochasticity comes from sampling $\tilde{X}_0^{(t)} \sim \tilde{p}_\theta(x_0|y^{(t)})$ in the DDRM and the transition kernels are:

$$K^{[lb]}(\hat{X}_0^{(1:t-1)}, A) = \mathbb{P}\Big(\mathrm{GCN}^{[lb]}(\tilde{X}_0^{(t)} \sim \mathrm{DDRM}(\tilde{\mu}_\theta^{[lb]}, y^{(t)})) \in A\Big), \tag{33}$$

where $\mathrm{GCN}^{[lb]}$ and $\tilde{\mu}_\theta^{[lb]}$ are trained on the previous lb days.

Under the assumption we re-initialise parameters before training on day t we have:

$$\mathbb{P}\Big(\hat{X}_0^{(t)} \in A \,\Big|\, \hat{X}_0^{(t-1)}, ..., \hat{X}_0^{(1)}\Big) = \mathbb{P}\Big(\hat{X}_0^{(t)} \in A \,\Big|\, \hat{X}_0^{(t-1)}, ..., \hat{X}_0^{(t-lb)}\Big), \tag{34}$$

thus $(\hat{X}_0^{(t)})_{t\geq1}$ is a Markov chain with memory lb. We now state the relevant ergodic Theorem which is key to our main result; Theorem 3.

Theorem 2. *Let $(Z_t)_{t\geq1}$ be an ergodic Markov chain with (possibly infinite) state space Ω. If μ is the equilibrium distribution of*

$(Z_t)_{t\geq 1}$ then for any initial distribution:

$$\lim_{n\to\infty}\frac{1}{n}\sum_{t=1}^{n} f(Z_t) \overset{a.s.}{=} \int_{\Omega} f\, d\mu\,, \tag{35}$$

for all μ-integrable functions $f:\Omega\longrightarrow\mathbb{R}$ with $\int_\Omega |f|\,d\mu<\infty$.

Proof. Theorem 3 from Tierney's paper [25]. □

The other result we require is the following:

Lemma 1. *Assume $lb = 1$ and $S_{q(x_0)} \subseteq Im(\text{GCN})$, that is the support of $q(x_0)$ lies within the GCN range. Then if $\widetilde{\mathcal{D}}^{-\frac{1}{2}}\widetilde{\mathcal{W}}\widetilde{\mathcal{D}}^{-\frac{1}{2}}$, $\Theta^{(0)},...,\Theta^{(L-1)}$ and σ from Eqn (17) are invertible it follows that $(\hat{X}_0^{(t)})_{t\geq 1}$ is strongly irreducible with respect to $q(x_0)$.*

Proof. For $lb = 1$ Eqn (34) tells us that $(\hat{X}_0^{(t)})_{t\geq 1}$ is a Markov chain with transition kernel:

$$K^{[1]}(\hat{X}_0^{(t-1)}, A) = \mathbb{P}\Big(\text{GCN}^{[1]}(\tilde{X}_0^{(t)} \sim \text{DDRM}(\tilde{\mu}_\theta^{[1]}, y^{(t)})) \in A\Big)\,, \tag{36}$$

$\forall A\in\mathcal{B}_\Omega$ by Eqn (33). From Gaussian sampling in the DDRM we have a sample space $\Omega=\mathbb{R}^{n\times d}$ and thus $\forall t\geq 1$:

$$P\big(\tilde{X}_0^{(t)} \sim \text{DDRM}(\tilde{\mu}_\theta^{[1]}, y^{(t)}) \in A\big) > 0\,, \tag{37}$$

$\forall A \in S_{q(x_0)}$, $\forall \hat{x}_0^{(t-1)} \in \Omega$, where $\hat{x}_0^{(t-1)}$ is the past sample used for training. Note Eqn (17) gives:

$$\text{GCN}(X) = f^{(L-1)}(\cdots f^{(0)}(X))\,, \tag{38}$$

where:

$$f^{(l)}(X) := \sigma(\widetilde{\mathcal{D}}^{-\frac{1}{2}}\widetilde{\mathcal{W}}\widetilde{\mathcal{D}}^{-\frac{1}{2}}\, X\,\Theta^{(l)})\,, \tag{39}$$

which from the assumed invertability has associated inverse:

$$g^{(l)}(X) := \widetilde{\mathcal{D}}^{\frac{1}{2}}\widetilde{\mathcal{W}}^{-1}\widetilde{\mathcal{D}}^{\frac{1}{2}}\,\sigma^{-1}(X)\,(\Theta^{(l)})^{-1}\,, \tag{40}$$

$\forall X\in g^{(l+1)}(\cdots g^{(L-1)}(Im(\text{GCN})))$. Thus $\forall A\in Im(\text{GCN})$:

$$\mathbb{P}\Big(\text{GCN}^{[1]}(\tilde{X}_0^{(t)})\in A\Big) = \mathbb{P}\Big(\tilde{X}_0^{(t)}\in g^{(0)}(\cdots g^{(L-1)}(A))\Big) \tag{41}$$

$$> 0\,, \tag{42}$$

where line 42 follows from Eqn (37). Finally, as $S_{q(x_0)}\subseteq Im(\text{GCN})$ the result now follows. □

Remark: The $lb = 1$ case can practically be achieved for the DDPM, see our SinDDM comments in Section 5 [14]. We leave the generalisations of this argument to chains with memory ($lb > 1$) for future work.

We now have our main result:

Theorem 3. *If the assumptions of Lemma 1 hold and $(\hat{X}_0^{(t)})_{t\geq 1}$ is positive Harris recurrent with invariant distribution $q(x_0)$ then Assumption 2 holds as $t\longrightarrow\infty$.*

Proof. From Lemma 1 $(\hat{X}_0^{(t)})_{t\geq 1}$ is strongly irreducible thus we immediately have that $(\hat{X}_0^{(t)})_{t\geq 1}$ is irreducible and aperiodic. Thus, as $q(x_0)$ is assumed to be invariant, it follows by Theorem 1 in the work of Tierney that $q(x_0)$ is the equilibrium distribution for $(\hat{X}_0^{(t)})_{t\geq 1}$ [25].

Furthermore as we assumed $(\hat{X}_0^{(t)})_{t\geq 1}$ to be positive Harris recurrent it follows that $(\hat{X}_0^{(t)})_{t\geq 1}$ is ergodic. Now $\forall A\in\mathcal{B}_\Omega$ we

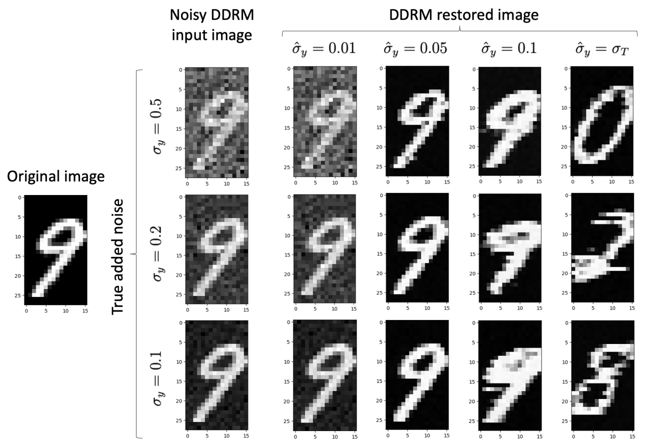


Figure 3: Different $\hat{\sigma}_y$ values being trailed to mimic σ_y being unknown in a DDRM where $\mu_\theta(\cdot,\cdot)$ comes from a PIU-Net. Here a DDPM was trained on the MNIST data set and the original image was an unseen sample [5].

define $f(x) := \mathbb{1}\{x\in A\}$, which is μ-integrable with $\int_\Omega |f|\,d\mu = \mu(A)\leq\mu(\Omega)=1$. Thus noting:

$$\int_\Omega f\,dq(x_0) = \mathbb{E}_{X\sim q(x_0)}[\mathbb{1}\{X\in A\}] = \mathbb{P}_{X\sim q(x_0)}(X\in A)\,, \tag{43}$$

Theorem 2 gives:

$$\lim_{n\to\infty}\frac{1}{n}\sum_{t=1}^{n}\mathbb{1}\{(\hat{X}_0^{(t)}|y^{(t)})\in A\} \overset{a.s.}{=} \mathbb{P}(X\in A)\quad \forall A\in\mathcal{B}_\Omega\,, \tag{44}$$

where $X\sim q(x_0)$, the result now follows. □

Remark: We believe the assumption that $q(x_0)$ is invariant should hold as during training the model would not benefit by deviating from it. Thus setting learning rates carefully to avoid overshooting minima will be crucial.

Theorem 3 shows that under the right conditions, our iterative GCN, DDPM, DDRM method can be seen as an MCMC approach to learning and sampling data from the true underlying distribution. As is the case with many MCMC methods, a burn-in period is likely beneficial as initially Assumption 2 will be a poor approximation.

An issue that arises from not knowing the true distribution is the σ_y value in Assumption 1 is unknown despite being needed for the DDRM. Letting $\hat{\sigma}_y$ denote the σ_y value used in denoising we find the DDRM to be robust to changes in σ_y for $\hat{\sigma}_y = 0.05$. Kawar et al. also use $\sigma_y = \hat{\sigma}_y = 0.05$ for their denoising examples [11]. Figure 3 shows various $\hat{\sigma}_y$ being trialed for different σ_y values on an application to the MNIST data set purely for illustrative purposes [5]. Here $\hat{\sigma}_y = 0.05$ has strong sampling results even as we range σ_y from 0.1 to 0.5 (where the input image pixels range is $[-1, 1]$).

4 APPLICATION TO FINANCIAL DATA

4.1 Financial graph construction

For testing our GDN framework and benchmark comparison, we focus on an equities return application. As mentioned in Section 1, we view this as a highly applicable domain since the data is noisy, has an underlying signal and originates from multiple sources. Furthermore, both assumptions should hold as while Theorem 3 proves Assumption 2 holds more generally, as argued in Section 3 financial returns data is a specific example that should also satisfy Assumption 1. We build our financial network as a "dynamic graph temporal signal" object to allow use of the PyTGT data compiler [19]. Our graph consists of 200 vertices, each corresponding to a US equity. For our vertex features we use daily returns, specifically beta neutral log close-to-close returns defined as:

$$r(S_t) := \log\left(\frac{S_t}{S_{t-1}}\right) - \log\left(\frac{\mathrm{SPY}_t}{\mathrm{SPY}_{t-1}}\right), \tag{45}$$

where S_t corresponds to the stock S close price on day t, and SPY is used to hedge (taking $\beta = 1$), ensuring dollar neutral portfolios. For our graph weights, we use the Wharton Research Data Services (WRDS) individual stock fundamentals data. For each day, we apply a PCA transformation across the 230 WRDS features, retaining the first p principal components that make up $\geq 80\%$ of the variance, with the purpose of denoising the data. Next, we compute the distance correlation for the post-PCA vectors of each vertex pair. Finally, we set any correlations below 0.7 to 0, leaving us with $20\% - 30\%$ of all possible edges.[3]

4.2 Loss function

Our model has two separate loss functions. The first is the unsupervised loss as in Eqn (24) for our DDPM module. The second is the main GDN loss where we train in a supervised setting. As we are looking at equity returns, we take inspiration from Vuletić et al. [27] who introduced a loss combining MSE, profit and loss (PnL) and Sharpe Ratio (SR). We build our loss around MSE and PnL, similar to an elastic net approach. Defining $S_t^\alpha(\theta)$ as the models prediction of $r(S_{t+1})$ for parameter θ, the PnL on day t is:

$$\mathrm{PnL}_t(\theta) := \frac{1}{200}\sum_{S=1}^{200} \mathrm{sign}(S_t^\alpha(\theta)) \times r(S_{t+1}). \tag{46}$$

An issue here is $\frac{d}{dx}\mathrm{sign}(x) \overset{a.s.}{=} 0$, thus PnL_t can not be used for stochastic gradient descent (SGD). To overcome this, we follow Vuletić et al. [27] and approximate PnL_t with:

$$\widetilde{\mathrm{PnL}}_t(\theta, \gamma) := \frac{1}{200}\sum_{S=1}^{200} \tanh(\gamma S_t^\alpha(\theta)) \times r(S_{t+1}), \tag{47}$$

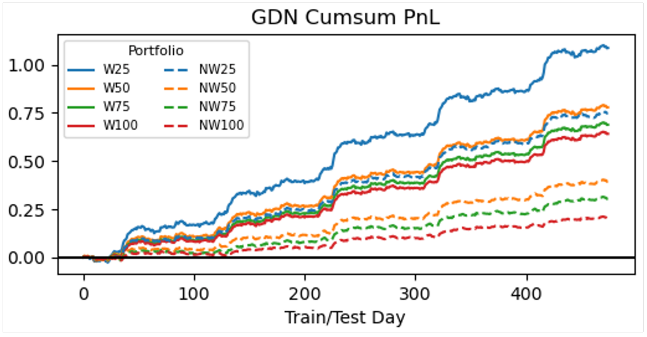
where γ controls the gradient of $\tanh(\gamma x)$ around $x = 0$ [27]. Our final GDN loss is then given by:

$$L_{\delta,\gamma}(\theta) := \delta(-\widetilde{\mathrm{PnL}}_t(\theta,\gamma)) + (1-\delta)\mathrm{MSE}(S_t^\alpha(\theta), r(S_{t+1})). \tag{48}$$

4.3 Results and discussion

For our train/test regime, we consider periods of 100 days, which allows us to have a rolling validation set; we use the last 100 days of train/test results to tune the hyper-parameters for the next 100 days. Our full backtest is over 2 years of trading data, and on any single day we train on the last 5 days and then predict the current day. We consider non-weighted portfolios (NW), where every trade is of equal size as in Eqn (46), and weighted portfolios (W), where the size of each trade is proportional to the size of the signal $S_t^\alpha(\theta)$.

[3]Distance correlation takes values in $[0, 1]$ thus we need not consider negative weights.



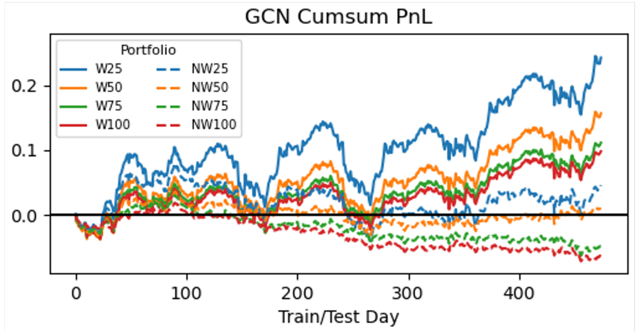


Figure 4: The cumsum PnL for the GDN (top) and GCN (bottom) over the eight portfolios.

We also consider quantile portfolios, where we only trade on the $Q\%$ of strongest signals for $Q = 25, 50, 75, 100$.

Figure 4 (top) shows the cumulative sum (cumsum) PnL for the GDN under these eight portfolios. Here, we observe that the weighted portfolios outperform their non-weighted counterparts, and performance is inversely proportional to Q, both as expected. The GDN shows very strong performance with all portfolios having an annualised SR above 2.23 and the strongest strategy, W25, having a SR of 4.41. To more rigorously conclude that the GDN produces positive Sharpe Ratios, we use the hypothesis test introduced by Bailey et al. that corrects Sharpe Ratios for selection bias under multiple testing and non-Normally distributed returns [1]. Testing $H_0 : \mathrm{SR} = 0$ vs $H_1 : \mathrm{SR} > 0$, we obtain p-values of $p < 1 \times 10^{-4}$ for all eight portfolios, giving strong evidence to reject H_0 in favour of H_1, and concluding that the GDN produces positive Sharpe Ratios.

To highlight the additional performance the GDN yields, we compare it to the baseline GCN to directly see any improvements. Figure 4 (bottom) shows the cumsum PnL for the GCN over the same period. Here we see two of the portfolios now produce a loss, while the strongest portfolio, again W25, returns only a quarter of the GDN W25 portfolio. Figure 5 further highlights the discrepancy in model performance; here, we see the GDN greatly outperforms the GCN across all eight portfolios for annualised SR, annualised average return and PnL per trade. We note the GDN W25 portfolio has a very impressive annualised average return of 57.7% over the two years of testing.

To rigorously compare if the GDN PnL underlying mean, μ_{GDN}, is larger than the GCN PnL mean, μ_{GCN}, we use the two-sample t-test from Snedecor and Cochran [23]. Specifically we use the equal

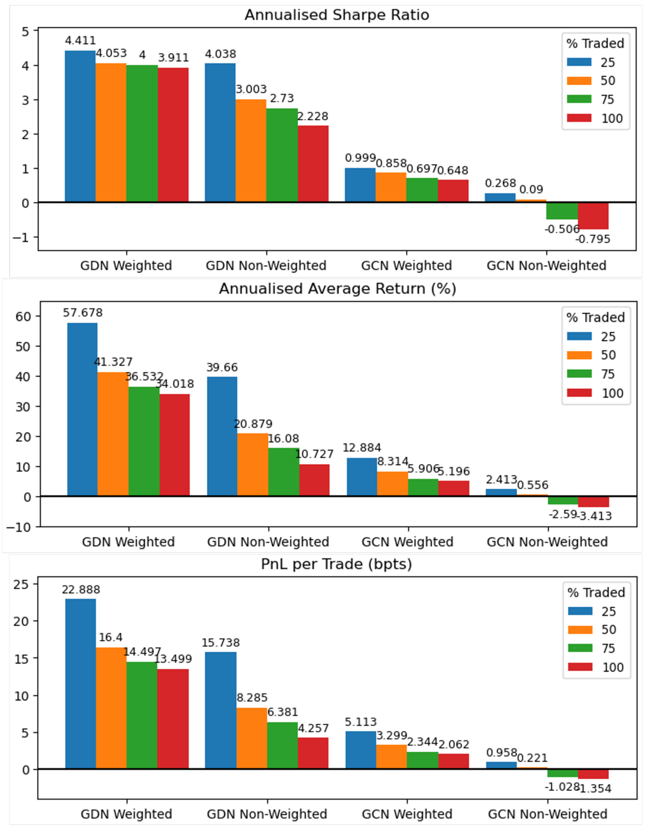


Figure 5: The annualised Sharpe Ratio (top), annualised average return (middle) and PnL per trade (bottom) for the GDN (left) and GCN (right) over the eight portfolios.

variances case, which we view as valid as each portfolio's GDN/GCN pairs have variances within 16%. We test: $H_0 : \mu_{\text{GDN}} = \mu_{\text{GCN}}$ vs $H_1 : \mu_{\text{GDN}} > \mu_{\text{GCN}}$, obtaining p-values of $p < 2 \times 10^{-4}$ for all eight portfolios, again giving strong evidence to reject H_0 in favour of H_1. Thus we conclude there is clear evidence that the GDN improves performance relative to the baseline GCN.

5 CONCLUSION AND FUTURE WORK

Our original aim was to produce a model that can handle noisy graph data in an attempt to extend GRL methods to such domains. To do this we introduced the novel GDN framework in Section 3 and provided the mathematical reasoning behind it in Lemma 1 and Theorem 3. Here we proved under the right assumptions it can be viewed as including a built-in MCMC method. We have shown in practice the model not only has strong results, with Sharpes consistently above 2.23 and yearly returns of up to 57.7%, but also outperforms the baseline GCN model. Our t-tests showed there is statistically significant evidence to conclude the GDN has a higher mean PnL than the GCN.

Our work opens up further avenues of investigation. One extension would be adapting the DDPM to SinDDM [14]. This would make the $lb = 1$ assumption in Theorem 3 hold as SinDDM provides a method to train a DDPM on a single input [14]. Another possible extension would be to adapt the model to Improved DDPM [16] which removes some restrictive assumptions and generalises the loss function, achieving noticeably better performance [16]. The reason we do not build our GDN around Improved DDPM is that it invalidates Theorem 1, however this may be minor upon testing.

To add further validity to our results, future research could involve changing the graph component. We used a GCN as it is interpretable and Assumption 1 is applicable to many domains; however, this could be adapted to other architectures such as graph attention networks (GATs) [26]. Doing this may provide performance increases and comparing GAT to a GAT-based GDN could provide further evidence to support our findings.

ACKNOWLEDGMENTS

ET gratefully acknowledges financial support from the Oxford-Man Institute of Quantitative Finance, University of Oxford, that made this research possible.

REFERENCES

[1] David H. Bailey and Marcos López de Prado. 2014. The Deflated Sharpe Ratio: Correcting for Selection Bias, Backtest Overfitting and Non-Normality. *Journal of Portfolio Management* 40 (5) (2014), 94–107. https://ssrn.com/abstract=2460551
[2] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2014. Spectral Networks and Locally Connected Networks on Graphs. arXiv:1312.6203
[3] John Y. Campbell, Andrew W. Lo, and A. Craig MacKinlay. 1997. *The Econometrics of Financial Markets.* Princeton University Press, Princeton, New Jersey.
[4] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. *NeurIPS* 29 (2016).
[5] Li Deng. 2012. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine* 29, 6 (2012), 141–142.
[6] William L. Hamilton. 2017. Graph Representation Learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning* 14, 3 (2017), 1–159.
[7] David K. Hammond, Pierre Vandergheynst, and Rémi Gribonval. 2011. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis* 30, 2 (2011), 129–150. https://doi.org/10.1016/j.acha.2010.04.005
[8] Mingguo He, Zhewei Wei, and Ji-Rong Wen. 2022. Convolutional Neural Networks on Graphs with Chebyshev Approximation, Revisited. arXiv:2202.03580
[9] Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising Diffusion Probabilistic Models. In *Advances in Neural Information Processing Systems*, Vol. 33. Curran Associates, Inc., 6840–6851.
[10] Konstantin Karchev, Noemi Anau Montel, Adam Coogan, and Christoph Weniger. 2022. Strong-Lensing Source Reconstruction with Denoising Diffusion Restoration Models. arXiv:2211.04365
[11] Bahjat Kawar, Michael Elad, Stefano Ermon, and Jiaming Song. 2022. Denoising Diffusion Restoration Models. In *Advances in Neural Information Processing Systems*, Vol. 35. Curran Associates, Inc., New Orleans, Louisiana, 23593–23606.
[12] Thomas N. Kipf and Max Welling. 2016. Semi-Supervised Classification with Graph Convolutional Networks. arXiv:1609.02907
[13] Thomas N. Kipf and Max Welling. 2016. Variational Graph Auto-Encoders. arXiv:1611.07308
[14] Vladimir Kulikov, Shahar Yadin, Matan Kleiner, and Tomer Michaeli. 2022. SinDDM: A Single Image Denoising Diffusion Model. arXiv:2211.16582
[15] Gustav Müller-Franzes, Jan Moritz Niehues, Firas Khader, Soroosh Tayebi Arasteh, Christoph Haarburger, Christiane Kuhl, Tianci Wang, Tianyu Han, Sven Nebelung, Jakob Nikolas Kather, and Daniel Truhn. 2022. Diffusion Probabilistic Models beat GANs on Medical Images. arXiv:2212.07501
[16] Alex Nichol and Prafulla Dhariwal. 2021. Improved Denoising Diffusion Probabilistic Models. arXiv:2102.09672
[17] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. 2022. Hierarchical Text-Conditional Image Generation with CLIP Latents. arXiv:2204.06125
[18] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015.* Springer International Publishing, Cham, 234–241.
[19] Benedek Rozemberczki, Paul Scherer, Yixuan He, George Panagopoulos, Alexander Riedel, Maria Astefanoaei, Oliver Kiss, Ferenc Beres, Guzman Lopez, Nicolas Collignon, and Rik Sarkar. 2021. PyTorch Geometric Temporal: Spatiotemporal Signal Processing with Neural Machine Learning Models. , 4564–4573 pages.

[20] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S. Sara Mahdavi, Rapha Gontijo Lopes, Tim Salimans, Jonathan Ho, David J Fleet, and Mohammad Norouzi. 2022. Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding. arXiv:2205.11487

[21] Ryosuke Sawata, Naoki Murata, Yuhta Takida, Toshimitsu Uesaka, Takashi Shibuya, Shusuke Takahashi, and Yuki Mitsufuji. 2022. A Versatile Diffusion-based Generative Refiner for Speech Enhancement. arXiv:2210.17287

[22] David I Shuman, Sunil K. Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. 2013. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine* 30, 3 (2013), 83–98.

[23] George W. Snedecor and William G. Cochran. 1989. *Statistical Methods, 8th Edition.* Iowa State University Press, Ames, Iowa.

[24] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. 2015. Deep Unsupervised Learning using Nonequilibrium Thermodynamics. In *Proceedings of the 32nd International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 37).* PMLR, Lille, France, 2256–2265.

[25] Luke Tierney. 1994. Markov Chains for Exploring Posterior Distributions. *The Annals of Statistics* 22, 4 (1994), 1701 – 1728. https://doi.org/10.1214/aos/1176325750

[26] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. arXiv:1710.10903

[27] Milena Vuletić, Felix Prenzel, and Mihai Cucuringu. 2023. Fin-GAN: Forecasting and Classifying Financial Time Series via Generative Adversarial Networks. https://ssrn.com/abstract=4328302

[28] Ling Yang, Zhilong Zhang, Yang Song, Shenda Hong, Runsheng Xu, Yue Zhao, Yingxia Shao, Wentao Zhang, Bin Cui, and Ming-Hsuan Yang. 2022. Diffusion Models: A Comprehensive Survey of Methods and Applications. arXiv:2209.00796

[29] Bing Yu, Haoteng Yin, and Zhanxing Zhu. 2018. Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence.* International Joint Conferences on Artificial Intelligence Organization, Stockholm, Sweden, 3634–3640. https://doi.org/10.24963/ijcai.2018/505

[30] Matthew D. Zeiler, Dilip Krishnan, Graham W. Taylor, and Rob Fergus. 2010. Deconvolutional networks. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition.* IEEE, Cape Town, South Africa, 2528–2535.