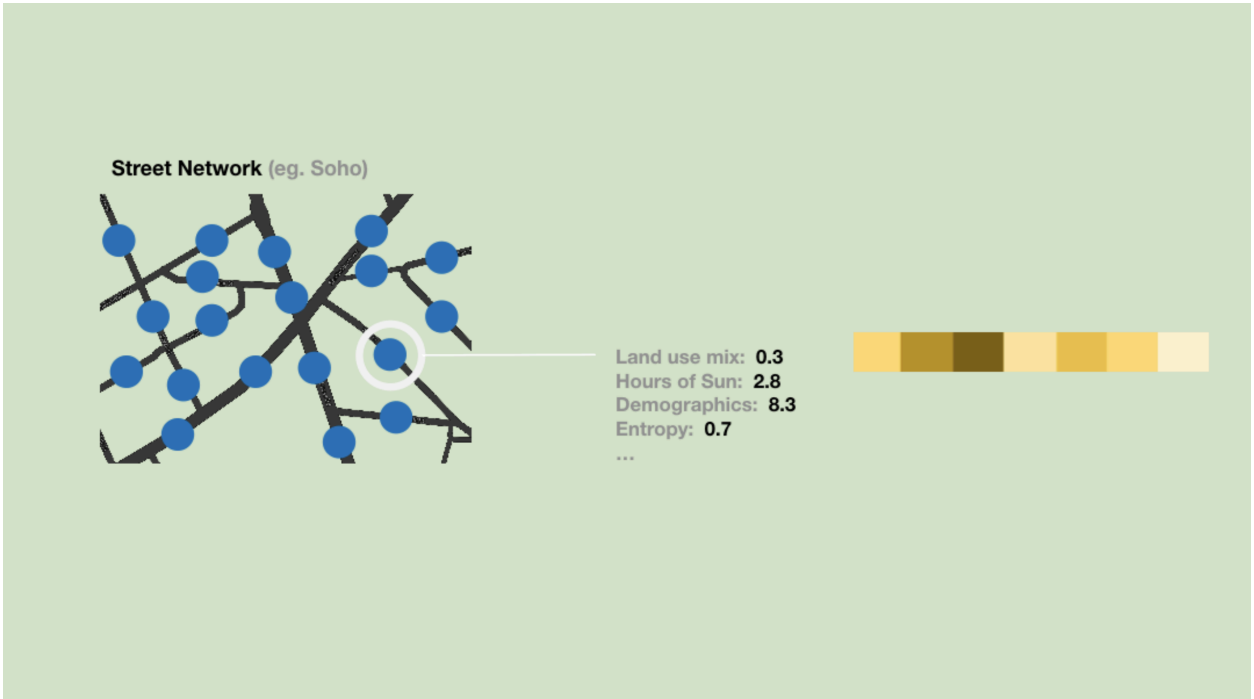
'Cities as Graphs' – Plotting value

Abstract

A house purchase is simultaneously a purchase of its aesthetic and structural features, its accessibility to work and friends, and the qualities of its neighbourhood. On the very local scale, its possible to measure the air quality, or even visual impression, of a given street. How the features of the wider neighbourhood are coupled with the value of a house is more difficult to quantify. Methods to create a 'neighbourhood fingerprint', by querying features within a given radius, miss essential information of how a street is connected into the wider network.

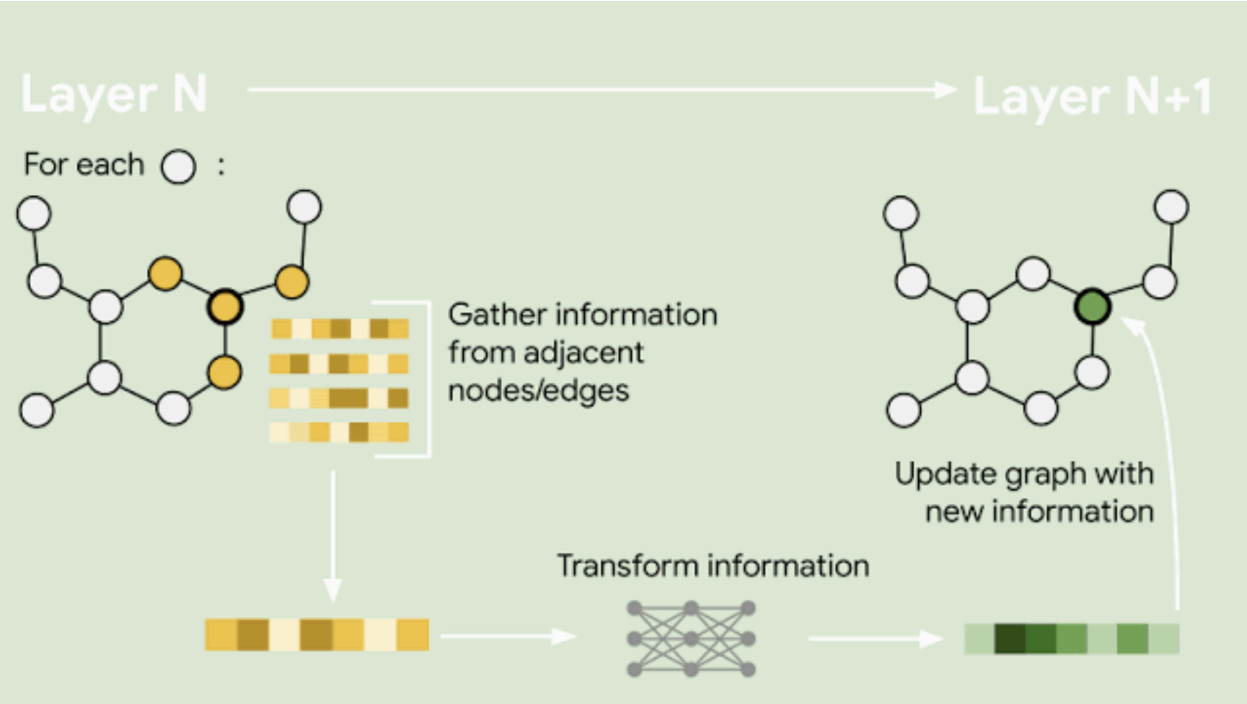
We propose that a novel class of machine learning architectures, known as graph neural networks, can more accurately delineate the relationship between the house's value, and the features of its vicinity, than traditional methods: variants of many layered perceptrons. This involves reconceptualising the street network as a graph, which is already a common practice in Space Syntax. The results appear to vindicate the approach, suggesting that the model can learn a notion of 'neighbourhood' in a way that makes it distinct from orthodox approaches that tend to treat houses as isolated entities, in vacuo. What's more, the model is flexible enough to accommodate rapid repurposing if, for instance, one was interested in 'quality of life' on a street, as opposed to house price.

1. Represent a street network as a Graph



Represent city neighbourhoods as graphs, where streets are nodes, and edges the connections between streets. Each node has a feature vector: each entry in the vector initially encodes some street-level information

2. Use Graph Neural Networks to help a node 'learn' about its surroundings



For each node we look at adjacent nodes and collect their information, which is then transformed with a graph neural network (GNN) into new information for the centred node. This procedure is performed iteratively. Other variants of graph neural networks (GNNs) utilise edge and graph-level information.

3. Use the new information to predict house prices on the neighbourhood or street level

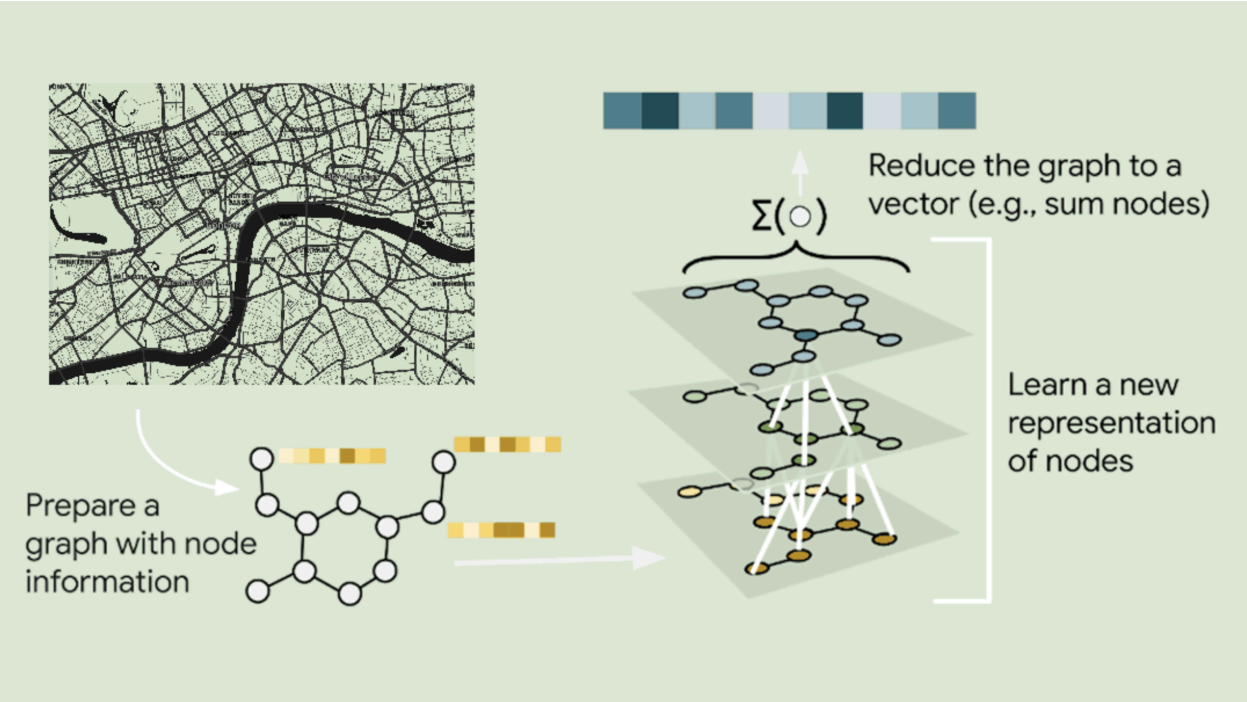
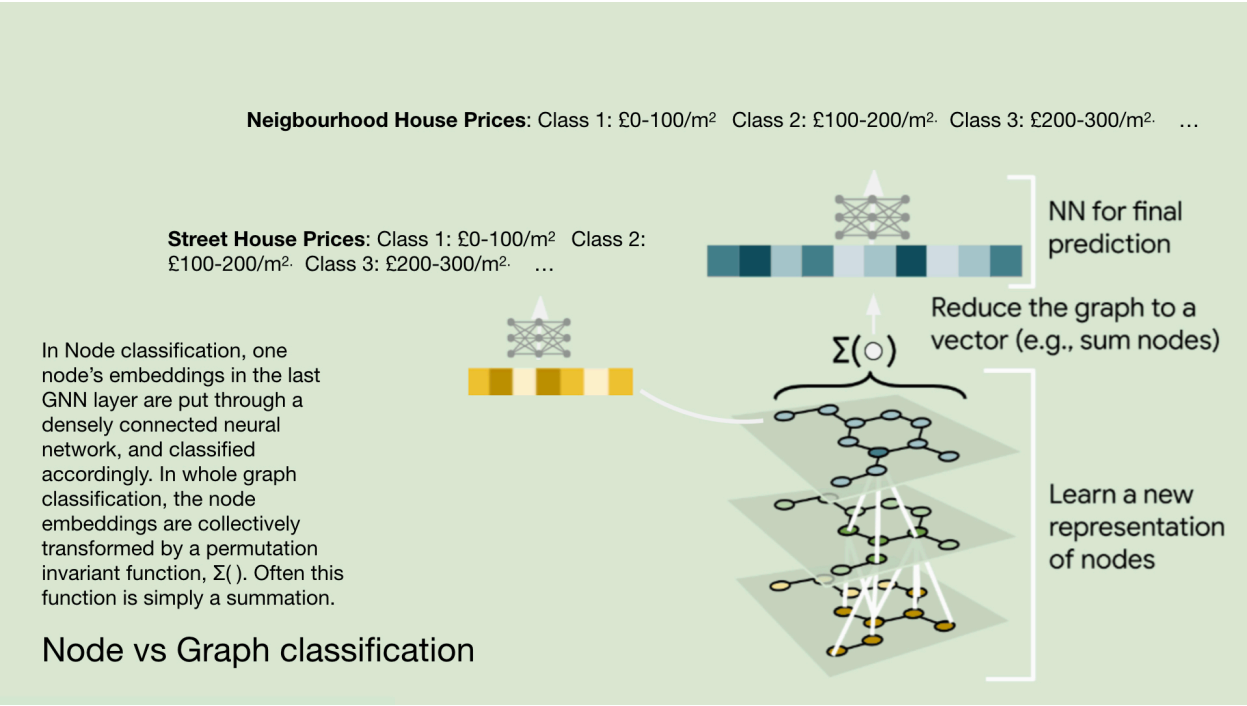


Illustration of a graph neural networks (GNNs) for street house price prediction. We translate the structure of neighbourhoods into graphs that are fed into GNN layers to learn a better representation of the nodes. These nodes can be reduced into a single vector to create an embedding representing the whole neighbourhood.

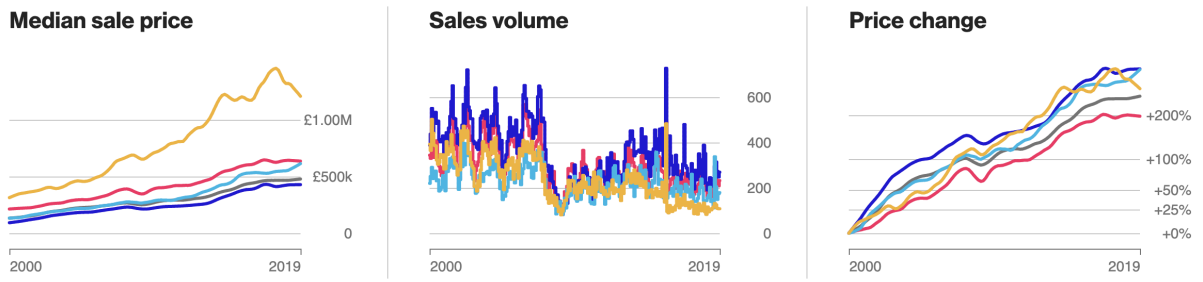


Individual node embeddings, or the whole graph embeddings, is passed through a dense neural network for classification purposes. (source: the images have been adapted from 'Learning to Smell: Using Deep Learning to Predict the Olfactory Properties of Molecules'. Alexander B Wiltschko et al.)

Problem definition

What makes a good place to live? For many, buying a house is the largest financial decision they will make in their lives. In recent history, the appreciation of house prices in the UK has outstripped wage increases. Equally, house price appreciation often cements economic inequality between different regions. London houses have increased in value at double the rate of those in Yorkshire, Wales or Scotland over the past 25 years. Even within cities themselves, the economic fortunes of inhabitants have diverged greatly. Buying a house in Lewisham has been almost twice as good an investment as buying a comparable house in Wimbledon over the past 30 years.

Demand for houses in particular areas arise often in the absence of manifest change in the actual housing stock, or features of an area. Often a dislocation in what you could term 'inherent value' (those areas which score highly by metrics of urban performance) and 'market value' are redressed as subsequent generations of house prices question the house buying orthodoxies of the previous generation. From anecdotal evidence, some postulate that in West London, for instance, the 'well to do' initially moved further westwards, before moving south of the river. Fulham became 'à la mode' as a cheaper version of the 'aspirational' location Chelsea. Subsequently Notting Hill, which had good housing stock — but had been imbued with the image of the All Saints Road race riots — became a go-to location, as well as Clapham. Similar narrative can be drawn for the movement out eastwards, although it is always challenging delineating the underlying causes of internal migration.



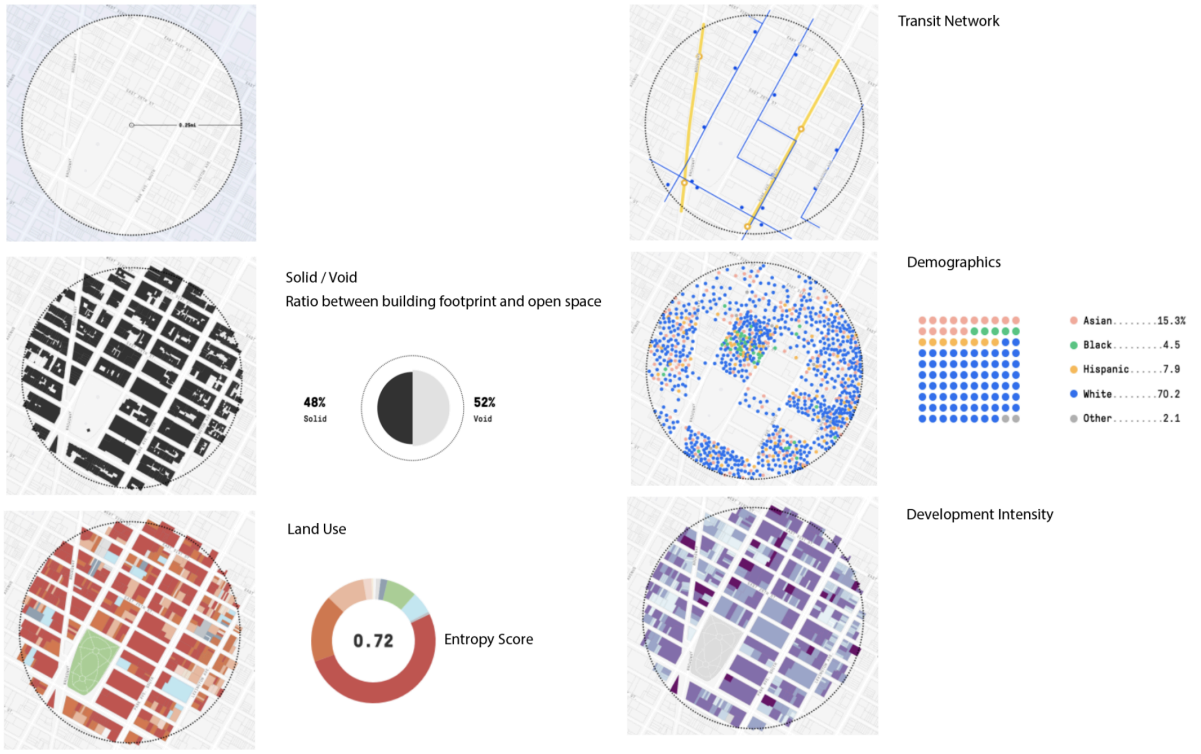
London median vs **Richmond upon Thames** vs **Lewisham** vs **Hackney** vs **Kensington and Chelsea**
(source Bloomberg)

This points something to a weakness in the way people purchase houses. For such a momentous decision, house purchasing continues to be done in a fairly luddite way. Those with sufficient capital will in the main choose properties by gut instinct, without a material understanding of how features of the chosen property are comparable with others across the area. The 'image' or 'perception' of an area often is a key driver of price, as opposed to what above we have termed 'inherent value'. As there is much less liquidity in the housing stock than in other 'asset classes', the rebalancing between the inherent and market value of housing stock occurs at a fairly ambling pace, and it will always be weighted by amorphous notions of 'prestige', or 'desirability'.

It seems of use to try and abstract away the influence of 'prestige', and comprehend how features of a locality correlate with its value. Of course, there has been a longstanding interest from commercial interest groups, such as developers, to do just that. There is a competitive advantage to understanding why particular subsets of the city become much more desirable to live within, from one epoch to the next — and, at even more granular a level, why a specific street is more attractive than those on its periphery. These analyses start with identifying a loose amalgam of metrics, which describe any given area. We may, for instance, notice that Hackney and Brixton became more interesting places to live as a consequence of the large artistic community that lived there: in other words, its cultural cachet. These artists themselves were perhaps drawn by economic necessity: affordable renting of studio space. The young are similarly drawn to cheaper locations, and — with growing wealth, and with there being a certain inertia against leaving a borough — they bootstrap the economic vitality of an area. As such, we might create datasets of age and professional profiles of people in different areas, in order to capture this insight. Those who work in Space Syntax are prone to hypothesising that you can discern from maps alone whether a particular area is likely to be inert, or thriving. [1] The assumption is that the built environment exercises a formative and even deterministic influence on the evolution of the societies that occupy it. Typically, for instance, the road network is analysed by measure of

‘choice’ and ‘integration’ to better understand a locality’s ‘entropy’: the experience of navigating the city for a pedestrian. [2] Clustering or graph partitioning algorithms are routinely used for to understand a city’s structure. [3] A heavily internally connected neighbourhood, which has sparse links to external neighbours, makes the remit of a resident more parochial, focusing everyday life more exclusively on one’s immediate neighbourhood. The structure of this network also results in its ‘serendipity’ (the measure by which you are likely to bump into other that you know). The land use mix, population density, number of dwelling units per land area, mobility, crime rate, and demographics are equally all obvious features of an area, which we may assume to be relevant to its value. These sit beside relatively unquantifiable properties, such as the ‘aesthetic’ of an area. How all these features work in concert, and how strongly they are coupled, is something of an enigma.

From the above it seems self evident that house price value is intricately tied to a large list of factors — and not least of all the features of its local neighbourhood. It would surprise no one to say that the fact that such and such pub, or this tube stop, or that particularly beautiful church within the remit of someone’s immediate vicinity makes a large different to quality of life. Standard regression analyses, however, which attempt to calculate a particular house price often forgets that houses are spatially located, and have a relationship with the environment they occupy. They treat houses or streets as in vacuo. The machine learning models upon which they are constructed are just not fit for purpose; their actual architecture is unable to compute the notion of a house or street's '5 minute walk' neighbourhood.



An illustration of the '5 minute walk' neighbourhood. (Source, https://explorer.morphocode.com/map)

There is, however, a potential solution in a novel set of machine learning architectures known as 'graph neural networks' (GNNs) [4]. These work on datasets that can be represented as 'graphs'. Graphs are mathematical structures made up of collections of 'nodes' and 'edges' connecting them. Each of the nodes and edges can be characterised by various features. How would one translate the structure of a city into a graph representation? One option is for the streets themselves to take on the role of nodes, and the connections between streets as edges. Initially, every node in the graph is then represented as a vector, using any preferred featurisation — percentage of residential housing versus office space, number of trees on street, etc. In a a series of message passing step, every node then 'broadcasts' its current vector value to its neighbours. An update function (learned during training) generates an updated vector value at each node, given the received 'messages'. The process can be repeated many times, and in so doing a single node gains oblique knowledge of nodes 'n-hops' away, where n is the number of message passing steps. The final node featurisations are much richer in knowledge than the initial ones. These can individually be passed into a fully connected neural network as a learned streets featurisation, whereby the network outputs a prediction for house prices. Equally, all of the nodes in the graph could be aggregated into a single vector via summing or averaging. That single vector, representing the entire neighbourhood, can then be passed into a fully connected network in a similar way

to do whole graph classification. The crucial point is that this methodology is able to 'understand' the notion of neighbourhood, and learn how the features of the whole may correlate with the features of a part.

> We will start with an overview of relevant academic research in the area, before exploring how cities can be represented as these 'graph' structures, and justifying from first principles why graph neural networks work. We will then detail how data was collected to input into our graph neural network model. Details of how to implement graph neural networks will be explored — although we will not discus all the possible variants of this class of ML models, which are very numerous. The next section will analyse our results, and disentangle what it reveals about the features which are most influential in determining the value of an area. Ultimately, the only thing which truly matters in machine learning is a methodology's efficacy. As such, we will conclude by benchmarking GNNs against other longstanding techniques of house price prediction.

Literature Review

Deep learning methods, as related to architecture and city analysis, have been largely restricted to a class of models known as 'convolutional neural nets'. Stanislas Chaillou, for instance, used pix2pix, a machine learning framework, to returns floorpans of a stylistic movement (baroque, Georgian etc.) based on supplied building footprints, and positions of apertures [5]. Ultimately one might ask: is this methodology, clearly appropriate for analysing 2D structure likes paintings, something of a sleight of hand when it comes to the 3D world of buildings, streets and architecture. Chaillou himself was preoccupied with ‘Architecture as graphs’: buildings and streets as most naturally represented as graph structures. Indeed, applying graph theory to city analysis is not novel. Space Syntax has commonly used a whole

host of graph theoretic tools. For these reasons, the intersection of deep learning and graphs theory seems to be the natural next progression of the field. There have been early attempt to apply the nascent field of graph neural networks (GNNs) to the systematic study of the built environment. Tinghua et al. used GNNs to classify building patterns in Wuhan [6]. That being said, most strides in combining graph theory and deep learning have come in the field of biochemistry. In ‘Machine Learning for Scent’, Alexander B Wiltschko et al. predicted the scent attributes of molecules by their underlying graph structure [7]. It illustrates that GNNs can predict emergent qualitative features from esoteric properties of an underlying graph network. Especially promising is the way in which the embeddings spaces produced by the GNNs had a hierarchical structure. Molecules with 'lavender', 'jasmine', or 'musguet' aromas inhabited a large cluster of 'floral' smells. There were similar subclasses for alcoholic and meaty smells. It gives hope that GNN embedding spaces, rather than just being an obscure repository of the higher dimensional representation of graphs, can actually provide straightforward insight, decipherable by the human eye. Much of the literature of GNNs is preoccupied with increasing the size of the graphs upon which the framework can feasibly operate [8]. Early instantiation of GNNs involved having to calculate the eigendecomposition of the graph network, which is heavily computationally expensive. Additionally, this eigendecomposition is only relevant to a particular graph structure: changing node or edge orderings will result in a completely different eigenfunctions of the graph. A plethora of different frameworks emerged to solve these twin problems.[9] Message passing neural networks became the de facto framework to unify the different flavours of GNN. [10]

House price prediction is a fairly common endeavour, and the literature is fairly extensive. As with successful environmental conservation, urban regeneration and restoration often depends directly or indirectly in how people assign values to various types of manmade and natural assets in the built environment. Having determined the value (monetary of otherwise) of property development, urban planners subsequently seek how to optimise areas for different environmental and social outcomes. Stephen Law has sought to understand, for instance, 'the intangible value of urban design.'[11] He hypothesises that urban aesthetics has a positive lasting effect on human health and general wellbeing, and indeed can influence people’s perception and decision-making associated with those spaces. His approach takes advantage of recent advances in citizen science and geo-data computation, analysing the collective perception of urban design with large volumes of crowd source data and machine learning techniques to infer the implicit economic value from the design of the built environment. This provokes questions as to the method by which we can collect data for machine learning on cities, and the data we should target. A key takeaway is the imperative to harvest free sources of data — Law used google streets maps images, which are not proprietary. Paying for data on the scale of a city it prohibitive, and the data collection process time consuming. Government open data is probably the most convenient, and richest source of data, but there may in future be other avenues.

Methodology

Mathematics of 'Deep Learning' on graphs

Machine learning is the process of finding a function to map an input to the 'correct' output.

In our case, we aim to predict an output 'label' from a graph input.

$$f : \{ \text{Graph} \} \rightarrow \{ \text{Label} \},$$

where {Graph} is the space of all graphs that could be constructed within our schema, and {Label} is a set of discrete labels describing features of the graph

There are six main 'flavours' of this, which determine the precise form of the output:

1. Node multi-label or binary classification
2. Node multi-label or binary classification
3. Node regression
4. Edge prediction
5. Graph multi-label of binary classification
6. Graph regression

The deep learning can additionally be done in either a supervised, unsupervised, or semi-supervised way. In the last instance, we train the model on graph where only a small subset of the nodes are labelled with ground truth information.

The function, f, is ill-defined, and operates on a domain of high dimension. As such, finding an exact construction of f is virtually impossible, and an approximation is sought. The first difficulty is that the 'space of graphs' is a high dimensional vector space, which can be spanned by many different chosen basis sets. Obviously a 'minimal spanning set' is preferred. Early methods involved manually attempting to construct a minimal spanning basis set by hand, guided by intuited assumptions about what constituted the essential characteristics of the graph.

Deep learning, by contrast, constructs models which are less theoretically complex, but more highly parameterised. Neural networks are a particular instance of such models. In their simplest form neural networks, NN_θ, are a composition of linear functions $f^l(.)$, indexed by $l \in 1, ..., L$, each followed by an element-wise non-linear function $\sigma(.)$:

$$NN_\theta = f^L \circ \sigma \circ f^{L-1} \circ \ldots \circ \sigma \circ f^2 \circ \sigma \circ f^1$$

Many NN variants have been proposed, differing mostly in the architecture of how certain elementary building blocks are combined in a computational graph, and how these building blocks are defined in the first place. For instance, in a 'many layered perceptron' (MLP), we define $f^l(.)$ as $f^l(X) = W_l X + b_l$, where W_l is the so-called weight matrix of the l^{th} layer and b_l is the bias vector. Both constitute the set of learnable parameters for a layer. We have used X to denote feature vectors (or hidden representations or embeddings) in a NN. Each individual MLP layer is simple and limited, but combining many such layers produces a structure capable of approximating many functions.

Our function, f, is likely to be of a very large number of parameters. $f(\theta_i)$. Generally we assume a 'weak prior' about the structure of the function. These assumptions (such as whether the function has certain symmetries) allows us to dramatically reduce the number of parameters involved. This dimensionality reduction facilitates the deployment of machine learning in situations where a vast number of parameters may make it otherwise infeasible. In other words: we tailor the function architecture to a particular problem domain. If certain properties of the machine learning problem are “built in, we can more successfully solve the targeted task. For instance, Convolutional neural networks are neural networks in which the linear map in some layers is replaced with a convolution . This produces layers of the form:

$$X_{(\ell+1)} = \phi\left(X_{(\ell)} * g_{(\ell)}\right)$$

where $g(\ell)$ is a convolutional 'kernel'. In discrete case of images, the convolutional kernel can be viewed as a small matrix of values, which is 'walked over' the input image, computing a two-dimensional analogue to the dot product at each location. Multiple kernels can be used simultaneously to process a set of input signals, and in so doing produce multiple output signals. These convolutional kernels are tensors — the higher dimensional analogue to matrices.

A Convolution operation can be mathematically expressed as:

$(f * g)(x) = \int_{\mathbb{R}^d} f(t) \cdot g(x - t) dt$, where $f, g : \mathbb{R}^d \to \mathbb{R}$ are real valued signals on a d-dimensional Euclidean space.

The convolution kernels, $g_{(\ell)}$, are produced in the same way as the parameters in the linear layers: that is, trained by gradient descent

The benefit of convolutional layers can be summarised as [12]:

1. **parameter reduction:** The size of these kernels can be explicitly controlled independently of the size of the input. Thus, the number of parameters in a single layer can be limited to an almost arbitrarily small number
2. **Localisation**: a particular output of a convolutional layer depends only on the values of a small set of neighbouring input values. For images and audio, for example, it is reasonable to expect that features of interest are local, and thus may be measured without considering values from across the entire input signal.

Typical datasets with which we use convolutional neural nets have a regular, Euclidean structure. Photos, for instance, are a regular grid of pixel values. Many 'modern' datasets, represented naturally be graphs, do not have such a regular structure.

The benefits of convolutions on Euclidean data sets, namely localisation and parameter reductions, are similarly desirable for problems defined on non-Euclidean domains. Additionally, it is reasonable to expect that such data sets might exhibit locality similar to that seen in images and audio. Information about users in a social network or papers in a citation network, for example, is likely more strongly dependent on close neighbours than on distant members. For our purposes, we would postulate that the neighbouring streets of a street network are relevant to the description of the central street. Given these considerations, it would be desirable to produce networks with benefits analogous to those of a standard convolutional neural network: locality and parameter reduction. However, the operation of convolution does not directly generalise to non-Euclidean domains and will require the development of a suitable analogy.

Extending the convolution operator to graphs

The generalisation of convolutional neural networks to non-Euclidean data sets such as graphs requires a convolution operator suitable for use over graphs. Such an operator is defined by analogy with the Fourier transform through the graph Laplacian matrix . The below proofs are quoted from Otness et. al. , and are given here for completeness. [13]

1. (Convolution Theorem). Convolution in the spatial domain corresponds to multiplication in the spectral domain $\widehat{(f * g)} = \hat{f} \cdot \hat{g}$

   ▼ proof

$$\begin{aligned}\widehat{(f*g)}(\xi) &= \int_{\mathbb{R}} (f*g)(t) e^{-2\pi i \xi t} dt \\ &= \int_{\mathbb{R}} \left(\int_{\mathbb{R}} f(u) \cdot g(t-u) du \right) e^{-2\pi i \xi t} dt \\ &= \int_{\mathbb{R}} \int_{\mathbb{R}} f(u) \cdot g(t-u) e^{-2\pi i \xi t} dt du \\ &= \int_{\mathbb{R}} f(u) \left(\int_{\mathbb{R}} g(t-u) e^{-2\pi i \xi t} dt \right) du\end{aligned}$$

where we changed the order of integration by application of Fubini's Theorem. Next we make a
change of variables with
$v = t - u$ so that $t = v + u$ and $dt = dv$

$$\begin{aligned}\int_{\mathbb{R}} f(u) \left(\int_{\mathbb{R}} g(v) e^{-2\pi i \xi (v+u)} dv \right) du &= \left(\int_{\mathbb{R}} f(u) e^{-2\pi i \xi u} du \right) \cdot \left(\int_{\mathbb{R}} g(v) e^{-2\pi i \xi v} dv \right) \\ &= \hat{f}(\xi) \cdot \hat{g}(\xi)\end{aligned}$$

2. (Relationship between Fourier transform and Laplacian operator): the Fourier Transform gives us a decomposition of a function f into a combination of eigenfunctions of the Laplacian operator, and under suitable restrictions these are also orthogonal

   ▼ proof

   Let $f : \mathbb{R}^n \to \mathbb{R}$. The Laplacian of f, denoted $\nabla^2 f$, is the sum of all second order partial derivatives of f : $\nabla^2 f = \sum_{\ell=1}^{n} \frac{\partial^2 f}{\partial x_\ell^2}$

   The complex exponentials, which appear in the fourier transform, are eigenfunctions of the Laplacian. In the single variable case, $\nabla^2 e^{-2\pi i \xi x} = -4\pi^2 \xi^2 e^{-2\pi i \xi x}$

   When we take the exponentials $e^{-2\pi i \xi}$ as elements of $L^2([0,1])$, the space of square integrable functions on the interval:
   $f \in L^2([0,1]) \Longrightarrow \int_0^1 |f(x)|^2 dx < \infty$

with inner product: $\langle f, g \rangle = \int_0^1 f(t) \cdot \overline{g(t)} dt$

and $\xi \in \mathbb{Z}$, then the exponentials are also orthonormal. This gives us values for the inner product of our “basis” functions

$$\begin{aligned} \xi, \xi' \in \mathbb{Z} \Longrightarrow \left\langle e^{-2\pi i \xi}, e^{-2\pi i \xi'} \right\rangle &= \int_0^1 e^{-2\pi i \xi t} \cdot \overline{e^{-2\pi i \xi' t}} dt \\ &= \int_0^1 e^{-2\pi i \left(\xi - \xi'\right) t} dt = \begin{cases} 1 & \text{if } \xi = \xi' \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

3. (Laplacian operator on graph): Euclidean Laplacian operator, ∇^2, can be generalised to L, the Laplacian operator in non-euclidean setting, where L = D - A (where D and A are the degree and adjacency matrix of the graph respectively).

▼ proof

In the Euclidean case for a function f , the Laplacian is the divergence of the gradient: $\nabla^2 f = \nabla \cdot \nabla f$

Consider real-valued functions on the set of the graphs vertices $\boldsymbol{f} : \mathcal{V} \longrightarrow \mathbb{R}$. Such a function assigns a real number to each graph node. The operations of gradient and divergence may be discretised. The gradient may be viewed as an operator taking a function on the vertices of the graph to a function on the edges, and the divergence may be viewed as an operator which gathers values from an edge function at the vertices.

$$\begin{aligned} (\nabla f_v)(i,j) &= \sqrt{W_{i,j}} \cdot (f_v(i) - f_v(j)) \\ (\nabla \cdot f_e)(i) &= \sum_{j:(i,j)\in E} \sqrt{W_{i,j}} \cdot f_e(i,j) \end{aligned}$$

where $f_v : V \to \mathbb{R}$ and $f_e : V \to \mathbb{R}$, $\nabla f_v : E \to \mathbb{R}$, $\nabla \cdot f_v : E \to \mathbb{R}$

These operators can be produced on the discrete graph space from the vertex-edge adjacency matrix, A.

$$A_{v,e} = \begin{cases} \pm\sqrt{W_{v,w}} & \text{if } e = (v,w), \text{ that is incident to } v \\ 0 & \text{otherwise} \end{cases}$$

The 'gradient' is therefore an application of A^T, and the divergence is an application of A.

$$
\begin{aligned}
(\nabla^2 f_v)(i) &= \sum_{j:(i,j)\in E} \sqrt{W_{i,j}} \cdot \nabla f_v(i,j) = \sum_{j:(i,j)\in E} W_{i,j}(f_v(i) - f_v(j)) \\
&= D_{i,i} f_v(i) - \sum_{j\in N_i} W_{i,j} \cdot f_v(j) = (L f_v)_i \\
&\implies AA^T = L
\end{aligned}
$$

We recover the graph Laplacian as a generalisation of the Euclidean Laplacian operator: $\nabla^2 \to L$

In quadratic form,
$\boldsymbol{f_v}^\top \mathbf{L} \boldsymbol{f_v} = \frac{1}{2} \sum_{e_{ij}} w_{ij} \left(f_v\left(v_i\right) - f_v\left(v_j\right)\right)^2$
which represents a smoothing of a signal over adjacent nodes.

4. From (2) we know the complex exponentials diagonalise the Euclidean Laplacian operator. Equally, we can diagonalise the graph Laplacian. $L = U^T \Lambda U$ where Λ is a diagonal matrix. One other point of similarity between the eigenbasis of the Graph Laplacian and the complex exponentials used in the Fourier transform is a loosely analogous notion of frequency. The Fourier transform decomposes a continuous signal into a combination of basic frequencies. In some sense the graph eigenvectors in order of increasing eigenvalue capture a similar notion of frequency

   ▼ proof

   For undirected graphs, L is symmetric real-valued matrices, and therefore has a complete orthonormal basis of eigenvectors.

5. (Graph Convolution) The graph Fourier transform of a signal $\mathbf{x} \in \mathbb{R}^n$ is defined as $\hat{x} = U^T x \in \mathbb{R}^n$, and its inverse as $x = \tilde{U}\hat{x}$

   ▼ proof

   Let $f, g : V \to \mathbb{R}$. Define $f * g = U^T(Uf \circ Ug)$, where $\odot$ denotes a pointwise multiplication, $(u \odot v)_i = u_i \cdot v_i$

The transformation U performs a change of basis on the vertex function, f_v , representing it as a linear combination of Laplacian eigenfunctions. This can be viewed as a discrete analogue of the Fourier Transform. We make use of this operation to define a convolution on a vertex fucntion by analogy with the covolution theorem.

Our construction of the fourier transform for graph respects the convilution theorem:

$\widehat{(f * g)} = U\left(U^T(Uf \circ Ug)\right) = Uf \circ Ug = \hat{f} \odot \hat{g}$

Graph Convolution Layer

From the previous section, we have established the following:

- The graph Laplacian, L, is a real symmetric positive semidefinite matrix. Consequently, it has a complete set of orthonormal eigenvectors, $\{u_l\}_{l=0}^{n-1} \in \mathbb{R}^n$, known as the graph Fourier modes, and an associated ordered set of real nonnegative eigenvalues, $\{\lambda_l\}_{l=0}^{n-1}$, identified as the frequencies of the graph.
- The graph Laplacian is diagonalised by the Fourier basis $U = [u_0, \ldots, u_{n-1}] \in \mathbb{R}^{n \times n}$, such that $L = U\Lambda U^T$, where $\Lambda = \operatorname{diag}([\lambda_0, \ldots, \lambda_{n-1}]) \in \mathbb{R}^{n \times n}$.
- The graph Fourier transform of a signal $\mathbf{x} \in \mathbb{R}^n$ is defined as $\hat{x} = U^T x \in \mathbb{R}^n$, and its inverse as $x = \tilde{U}\hat{x}$
- We can make use of the convolution theorem. Euclidean Convolution in the spatial domain corresponds to point multiplication in the spectral domain. That is, $\widehat{(f * g)} = \hat{f} \cdot \hat{g}$. Likewise, graph convolution is defined by: $\widehat{(f *_{\mathcal{G}} g)} = U\left(U^T(Uf \odot\ Ug)\right) = Uf \odot Ug = \hat{f} \odot \hat{g}$, where $\odot$ is defined by $(u \odot v)_i = u_i \cdot v_i$

From the above, we can define a graph convolutions layer in our neural network:

$X_{(\ell+1)} = \phi\left(U^T g_\Theta(\Lambda) U X_{(\ell)}\right),$

where U is the Graph Fourier Transform matrix, $g_\Theta(\Lambda)$ is the graph convolution kernel which is learned by gradient descent over its parameters Θ, ϕ is a non-linear function applied over the values of the signal and X are the inputs and outputs of the layer.

There are several issues with using this form of graph convolution layer in practice, the most obvious one being that the computational cost to diagonalise the Laplacian with standard algorithms scales as O(N3) which may quickly become prohibitive for very large graphs. [14] Furthermore, while L will be sparse for many real-world and constructed graphs, its Eigenbasis matrix U may be significantly more dense. We can reduce the parameter count of the convolution kernels by restricting the parametrisation. We do this by selecting a (small) set of basis kernels and producing spectral multipliers as a combination of these

In order to mitigate the issues to GCNNs outlined above, we choose to construct the convolution kernel g_Θ with particular basis functions:

Bruna et al. suggested using the smoothing filter [15]:

$\operatorname{diag}\left(\Theta_{i,j}^{l}\right)=\mathcal{K}\alpha_{l,i,j}$, , where $\mathcal{K}$ is a fixed interpolation kernel and $\alpha_{l,i,j}$ are learnable interpolation coefficients

Chebnet proposed to use polynomial filters [16]:

$\Theta(\lambda)=\sum_{k=0}^{K}\theta_k\lambda^k$

where $\theta_0,\ldots,\theta_K$ are the learnable parameters and K is the polynomial order

$$U^T g(\Lambda)U = U^T \begin{bmatrix} g\left(\lambda_0\right) & & \\ & & g\left(\lambda_{n-1}\right) \end{bmatrix} U = U^T \left(\sum_{\ell=0}^{k} a_\ell \Lambda^k\right) U$$
$$= \left(\sum_{\ell=0}^{k} a_\ell \left(U^T \Lambda U\right)^k\right) = g(L)$$

Kipf and Welling further simplified the filtering by using only the first-order neighbours [17]:

$\mathbf{x}_i^{l+1}=\phi\left(\sum_{j\in\tilde{\mathcal{N}}(i)}\frac{1}{\sqrt{\tilde{\mathbf{D}}(i,i)\tilde{\mathbf{D}}(j,j)}}\mathbf{x}_j^l\Theta^l\right)$

Equivalently, in matrix form:

$\mathbf{x}^{l+1}=\phi\left(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{x}^l\Theta^l\right)$

We will be using Kipf and Welling's formulation of graph convolutional layers, initially. Implementation details are discussed below. For analysis of very large graph networks, we will go on to use this implementation, developed by the All Data AI group at Microsoft Research, Cambridge, UK. [18]

Feature Choice

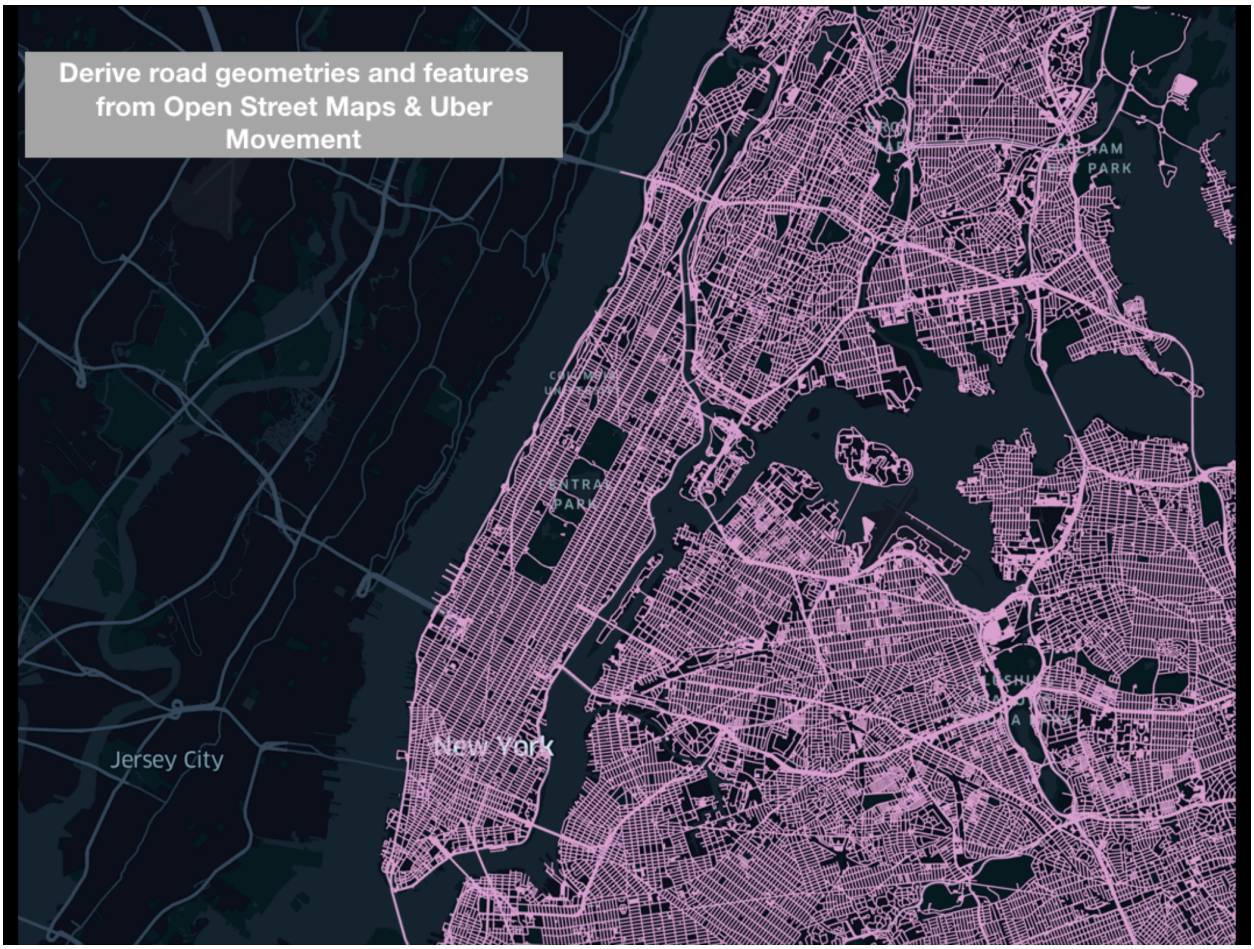
It is natural that we, as city dwellers, have a predisposition to viewing a particular property as essential to a city's vitality. Jane Jacobs, in 'The Life and Death of American Cities', makes a series of value judgements regarding the built environment:

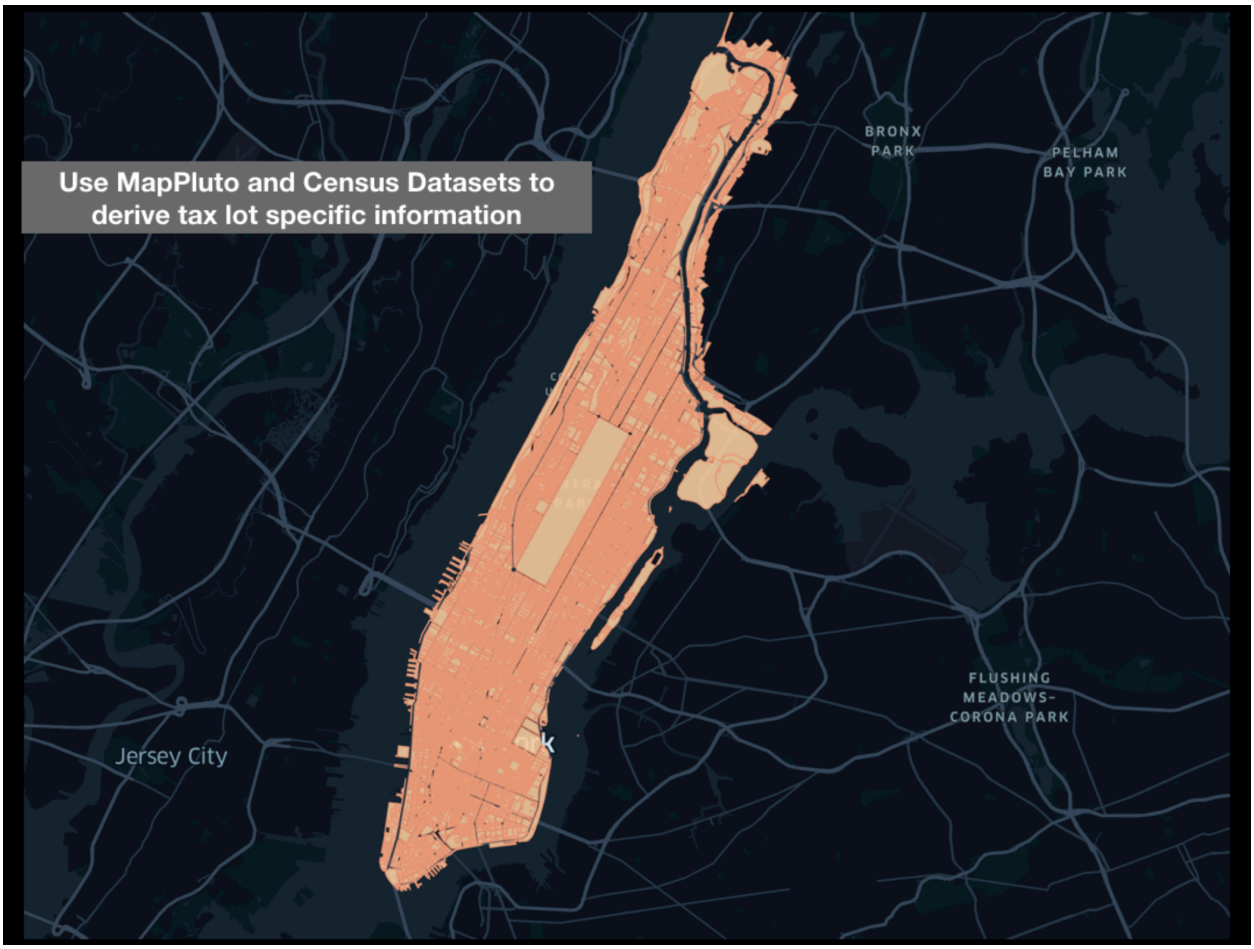
> To generate exuberant diversity in a city's streets and districts four conditions are indispensable:
> 1. The district, and indeed as many of its internal parts as possible, must serve more than one primary function; preferably more than two...
> 2. Most blocks must be short; that is, streets and opportunities to turn corners must be frequent.
> 3. The district must mingle buildings that vary in age and condition, including a good proportion of old ones so that they vary in the economic yield they must produce. This mingling must be fairly close-grained.
> 4. There must be a sufficiently dense concentration of people, for whatever purposes they may be there...

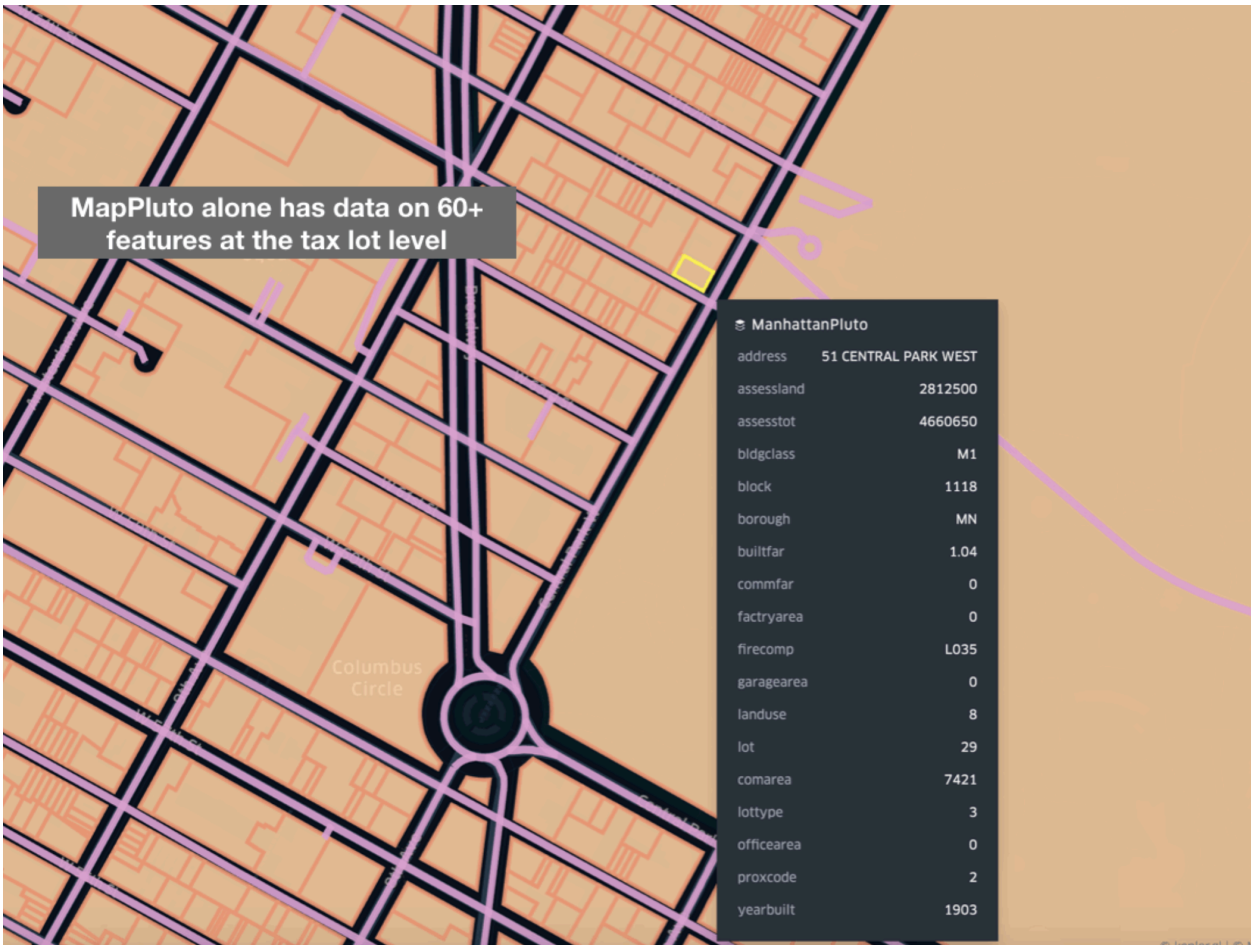
These insights were based off empirical observation — long periods spent surveying and stalking the sidewalks. Although they likely reveal a kernel of truth, it is difficult to know to what extent these observations reveal more about the watcher than the watched. Inevitably, influential voices set the tone for planning policy. Jane Jacobs is widely regarded, and her set of assumptions have permeated a generation of American planning decisions, for better or worse. GNNs potentially, however, can provide more direction to public planning. A GNN will learn to weight the importance of one feature against another — and how strongly various features are coupled. One can put to the test assumptions embedded in contemporary planning regulation. Of course, this initially involves creating models in which every conceivable feature of a city is accounted for. Features extraneous to the efficacy of the model can be can be discarded at a later date.

Below we have compiled a (by no means exhaustive) list of datasets describing features of New York City's built environment. NYC has been chosen for the quality and accessibility of its public datasets. With relative ease, we can use it as a laboratory to explore the effectiveness of GNNs in city analysis. The most useful of the datasets is MapPluto, which describes 60 features at the 'tax lot' level. The 2010 census data equally contributes a detailed account of the socio-economic status of residents in an area, soon to be updated by the 2020 census.

Datasets can be utilised in a variety of imaginative ways. While on face value Uber Movement data only documents the speeds, and trajectories of taxis through the city, it can also be used to infer information on noise pollution from cars. This can be cross referenced with the 311 service request data, which documents complaints noise complaints from other sources. Uber data can equally be used to make qualitative guesses at to the cultural or political beliefs on a street by street level — for instance, by tracking the endpoints of trips, whose source is NYC's pride festival. Some data — such as the amount of sunlight particular streets receive year-round — can be calculated using raytracing [19]. Microclimate data - as to the wind exposure, or heat island effect — can be calculated via fluid dynamics simulations. Ultimately data on 'intangibles', such as 'aesthetics' of a city street is difficult to come by. There has been some attempt to use semi-supervised learning to produce city-wide datasets of street beauty. Using google street views, humans by hand made value judgements on the beauty of streets in a subset of the city: the data was then extrapolated. If aesthetic appreciation is a form of pattern recognition, it is perhaps sufficient to describe the geometrical properties of streets: that is, the ratio of street length to building height, and so on..


Derive road geometries and features from Open Street Maps & Uber Movement
Jersey City
New York


Use MapPluto and Census Datasets to derive tax lot specific information
BRONX PARK
PELHAM BAY PARK
FLUSHING MEADOWS-CORONA PARK
Jersey City


MapPluto alone has data on 60+ features at the tax lot level
Columbus Circle
ManhattanPluto
address 51 CENTRAL PARK WEST
assessland 2812500
assesstot 4660650
bldgclass M1
block 1118
borough MN
builtfar 1.04
commfar 0
factryarea 0
firecomp L035
garagearea 0
landuse 8
lot 29
comarea 7421
lottype 3
officearea 0
proxcode 2
yearbuilt 1903

One of the main challenges is to aggregate data, which was compiled at different resolutions. The Pluto data is collected at the 'tax lot' level, whereas census data is anonymised over the blocks. This forces decisions as to on what level we construct our nodes/edges/graphs. It is possible for our graph nodes to represent individual tax lots, or collections of tax lots, or indeed whole blocks. Making sense of how we interpret the notion of 'edges' in each context is somewhat challenging.

For this study, we choose to view streets (i.e. collections of tax lots) as the nodes. Connections between streets can naturally be interpreted as graph edges. Information on the block level can then be viewed as features on the graph level, which can be used when doing graph classification or regression.

For our purposes, we chose to use only small number of potential features for the sake of simplicity.

yearBuilt: year houses were built on the street

refurbishment: time since houses on the street had last been refurbished.

bldgArea: amount of floor space of buildings on a street

bldgFront: the length of building exposed to the street for each house on the street

bldgDepth: the depth of buildings on the street

builtFar: ratio of buildings' total floor area (gross floor area) to the size of the piece of land upon which they were built.

numFloor: number of floors of buildings on the street,

comArea: an estimate of the exterior dimensions of the portion of the structures allocated for
commercial use on the street

resArea: an estimate of the exterior dimensions of the portion of the structures allocated for residential use on the street

factryArea: an estimate of the exterior dimensions of the portion of the structures allocated for
factory, warehouse or loft use on the street

retailArea: an estimate of the exterior dimensions of the portion of the structures allocated for
retail use on the street

garageArea: an estimate of the exterior dimensions of the portion of the structures allocated for
garage use on the street

strgeArea: an estimate of the exterior dimensions of the portion of the structures allocated for
storage or loft purposes on the street

lotArea: total area of the tax lots on the street

lotDepth: the tax lots' depth on the street

lotFront: the tax lots' frontage on the street

lotLandUse: ratio of floor space on the tax lots occupied by:

1. One & Two Family Buildings
2. Multi-Family Walk-Up Buildings
3. Multi-Family Elevator Buildings

4. Mixed Residential & Commercial Buildings
5. Commercial & Office Buildings
6. Industrial & Manufacturing
7. Transportation & Utility
8. Public Facilities & Institutions
9. Open Space & Outdoor Recreation
10. Parking Facilities
11. Vacant Land

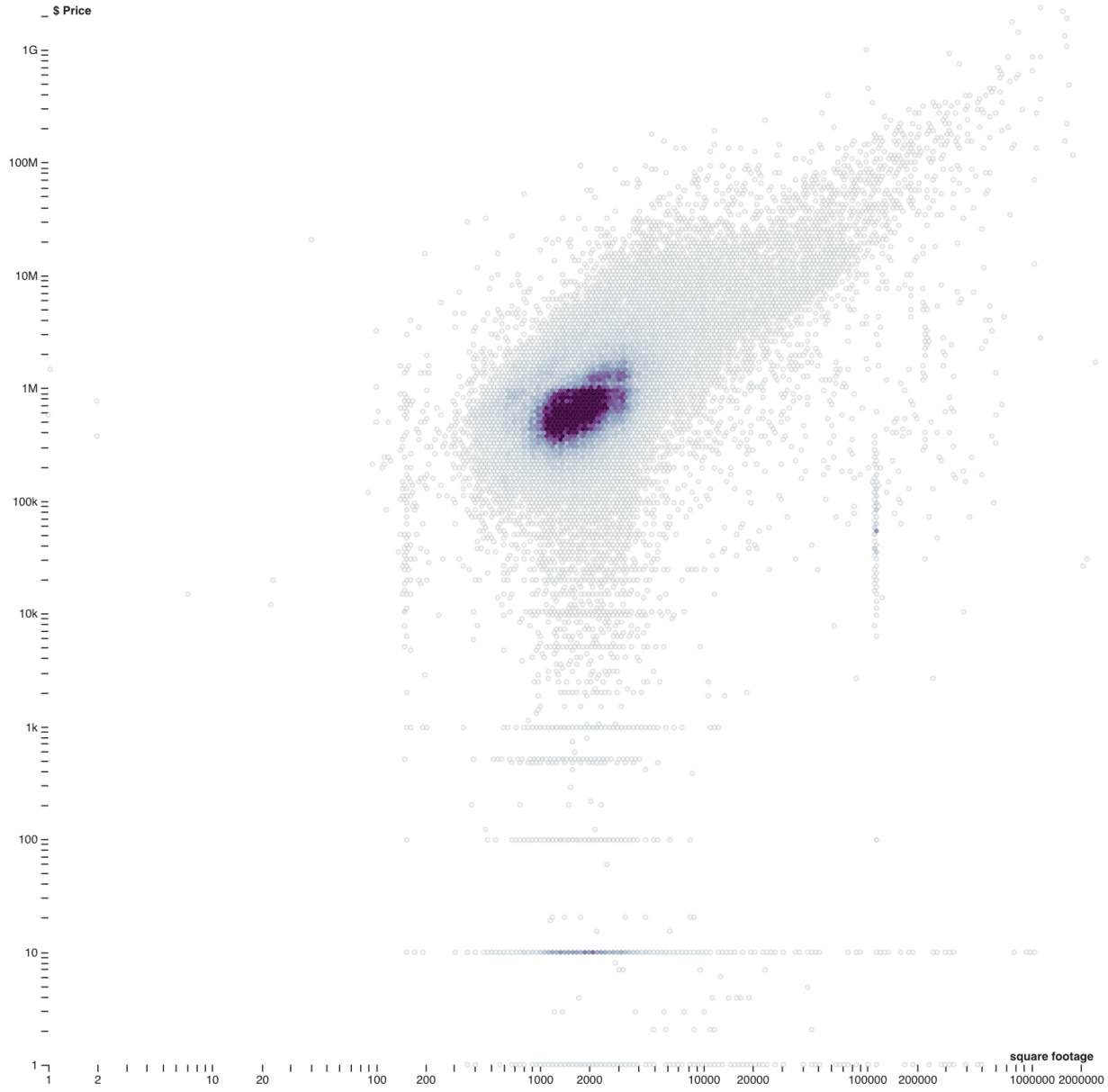
lotType: the location of the tax lots in relation to another tax lot and/or water

1. Unknown
2. Block assemblage – a tax lot that encompasses an entire block
3. Waterfront – a tax lot bordering on a body of water. Waterfront lots may contain a small amount of submerged land.
4. Corner – a tax lot bordering on two intersecting streets
5. Through – a tax lot connecting two streets, with frontage on both streets.
6. Inside – a tax lot with frontage on only one street. T
7. Interior lot – a tax lot that has no street frontage
8. Island lot – a tax lot that is entirely surrounded by water
9. Alley lot – a tax lot that is too narrow to accommodate a building. The lot is usually 12 feet or less in width.
10. Submerged land lot – a tax lot that is totally or almost completely submerged

Label Choice

Each place of residence in new york pays tax based on a property valuation assessment. We aggregated this data for each street, and assigned a value at the street level for average house price per square foot. We created 50 classes, which each represented an increment of price.

If we look at our data on property valuations against gross square footage, we notice that the range of valuations is between $0-1 Billion. The lots have a gross floorspace of between 100-2,000,000 sq. ft. The gross floorspace is a measure of both indoor and external floorspace of a tax lot. Our measure of value per sq. ft. is consequently going to be, in the main, less than that which is oft quoted for the price of property in New York. If we calculate the value of apartments by their internal floorspace alone, they are priced at $1,773/sq. ft., with a range of between $447/sq. ft. and $10,053/sq. ft. We, however, are summarising the total square footage of the building (including lobby areas, parking space, and outside land), and so value per square foot will be considerably lower.



Valuations of tax lots against their gross square footage in Manhattan

Label Class Numbers

Class Number	Average value of residential property on street per square foot
0	$0-200
1	$200-400
2	$400-800
..	
49	$10,000+

We then 1 hot encoded each label into 50-dimensional vectors.

Hence, for streets with an average property price of $53.30/square foot, the corresponding label, y, would be:

y = [0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0]

Implementation details

The above section illustrates how we processed 'street feature' vectors for each street in Manhattan. We next created a series of fully connected graphs of the streets in different neighbourhoods, and split these into training, validation and test sets. The 'test' set is to test the generalisability of our trained machine learning model, while the 'validation' set helps to prevent overfitting during the training of our machine learning model on the 'training' set.

We constructed our graph convolutional layers using the approach of Kipf and Welling:

$\mathbf{x}^{l+1} = \phi\left(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{x}^{l}\Theta^{l}\right)$, where D is the degree matrix, and A is the adjacency matrix.

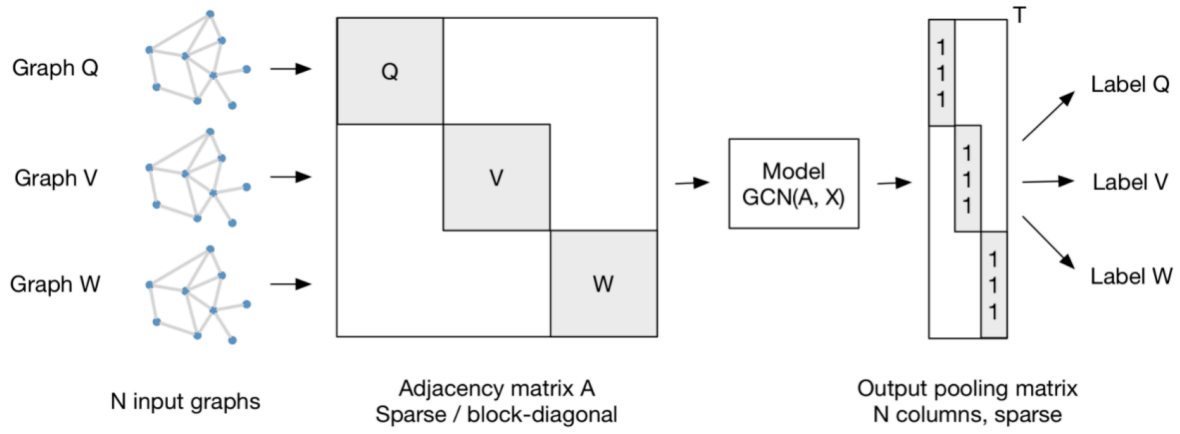
In order that a current node's features propagates to its next instantiation in the subsequent layer, we add 'self loops' in the degree and adjacency matrix, hence the notation $\tilde{\mathbf{D}}$ and $\tilde{\mathbf{A}}$. The pre- and post-multiplication by $\tilde{\mathbf{D}}^{-\frac{1}{2}}$ acts to normalise the adjacency matrix. In effect, this is a neural network layer in which nodes are connected with parametrised weights, $\Theta^{l}_{i,j}$, to nodes in subsequent layers, as determined by the graphs adjacency matrix. The parametrised weights are fine-tuned in order to learn how to propagate feature information from a node to its neighbours. After several of these layers, the feature vectors of each node will have a much 'richer' representation. The final node feature vectors can then be 'stacked' into a matrix, and put through a densely connected neural network for classification purposes. If both GCN and densely connected layers are put into a pipelined, and train against an output, the GCN layer will 'learn' to cluster node feature representations in this high dimensional vector space, otherwise known as an 'embedding space'. Nodes that will eventually be categorised similarly by the dense neural network layer will have a small Euclidean distance between each other in the embedding space. In the section below, attempts have been made to visualise this in action, by projecting the high dimensional embedding space in 3D.

There are two major difficulties to resolve at this point:

1. How do we train on graphs of different sizes?
2. How do we train these graphs in batches?

Obviously the first problem is only arises if we have more than one graph. In some situations (such as a citation network, or a graph of Bitcoin transactions), we only have one graph. A city, equally, could be represented as one large graph. In order that we can use our model on different neighbourhoods, in different cities we've chosen instead for our input graphs to be 'neighbourhood sized' graphs, which can be of different sizes, dependent on what we see as the natural size of a neighbourhood. Training in batches is useful only in that it greatly speeds up the process of training if we, for instance, have a training set of 10,000 'neighbourhoods'.

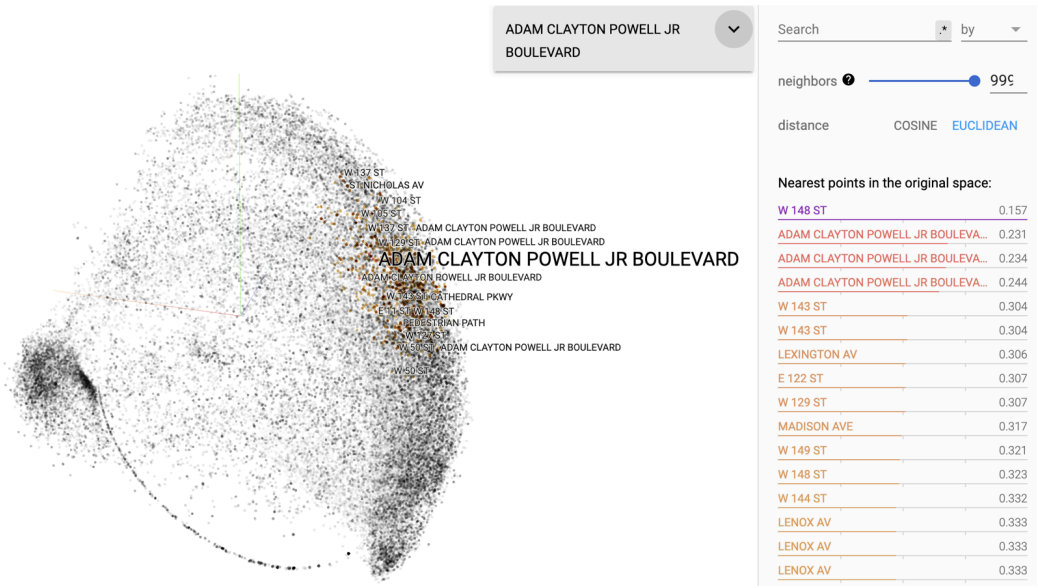
We resolve both these problems by setting a limit on the maximum size of our graph, and then pigeonholing all our graphs into this representation. As such, in graphs smaller than the maximum, we simply pad their adjacency and feature matrices with zeros. A similar solution can be utilised for batch training. We simply reconceptualise groups of graphs as one large graph of several unconnected subgraphs.



Method of doing batch training with graphs (source: graphic from Semi-Supervised Classification with Graph Convolutional Networks, Kipf and Welling).

Results & Analysis

Trained embeddings



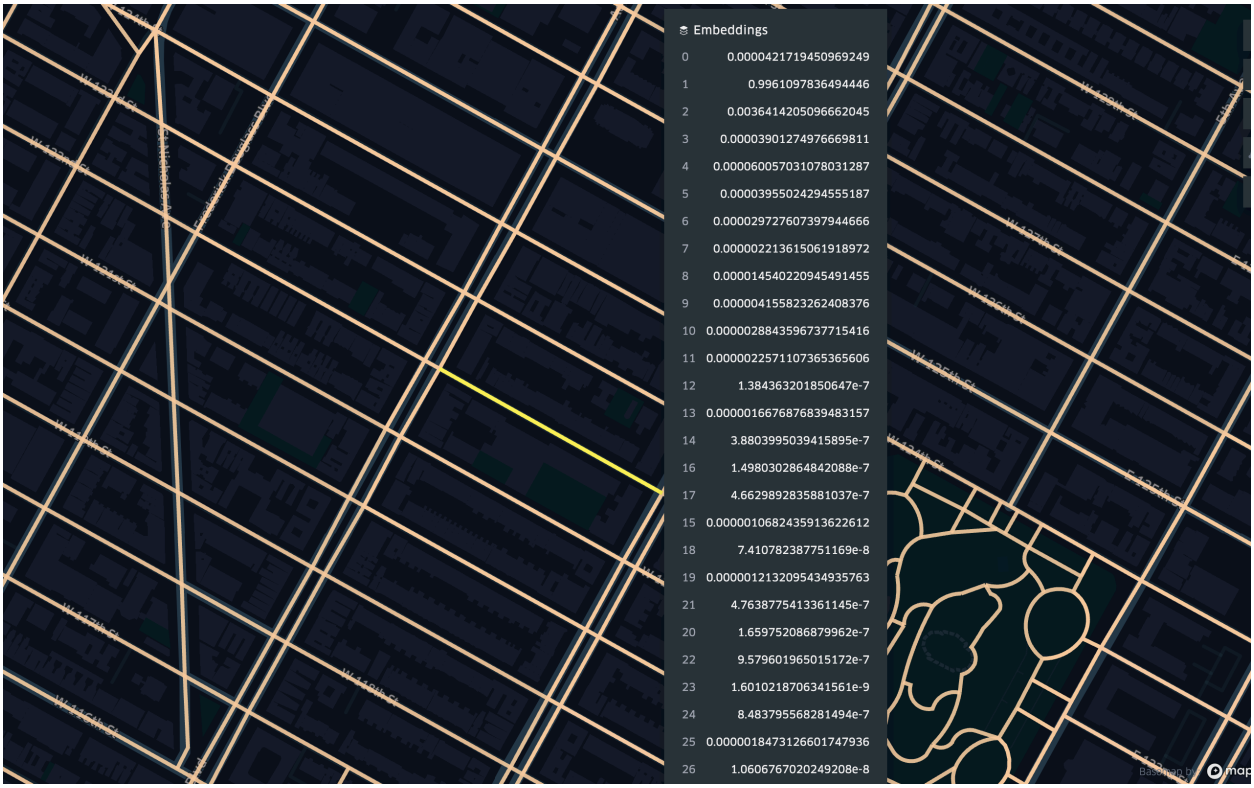
Visualisation of the trained embeddings in the laster GCN layer. In this instance, we are ascertaining which streets are most similar to 'Adam Clayton Powell Jr Boulevard'. The euclidean distance is the higher dimensional analogue to distance. We have split up streets into 'natural lengths' (those sections between intersections with another street'), but for simplicity have labelled them with their parent street name. We can interpret the clusters as streets which the model perceives to be of equal value. It may be the case that there is a large divergence between the nominal market value per square foot of properties on the streets in these clusters. That being the case, we may surmise that particular streets are either over- or undervalued.

Predictions

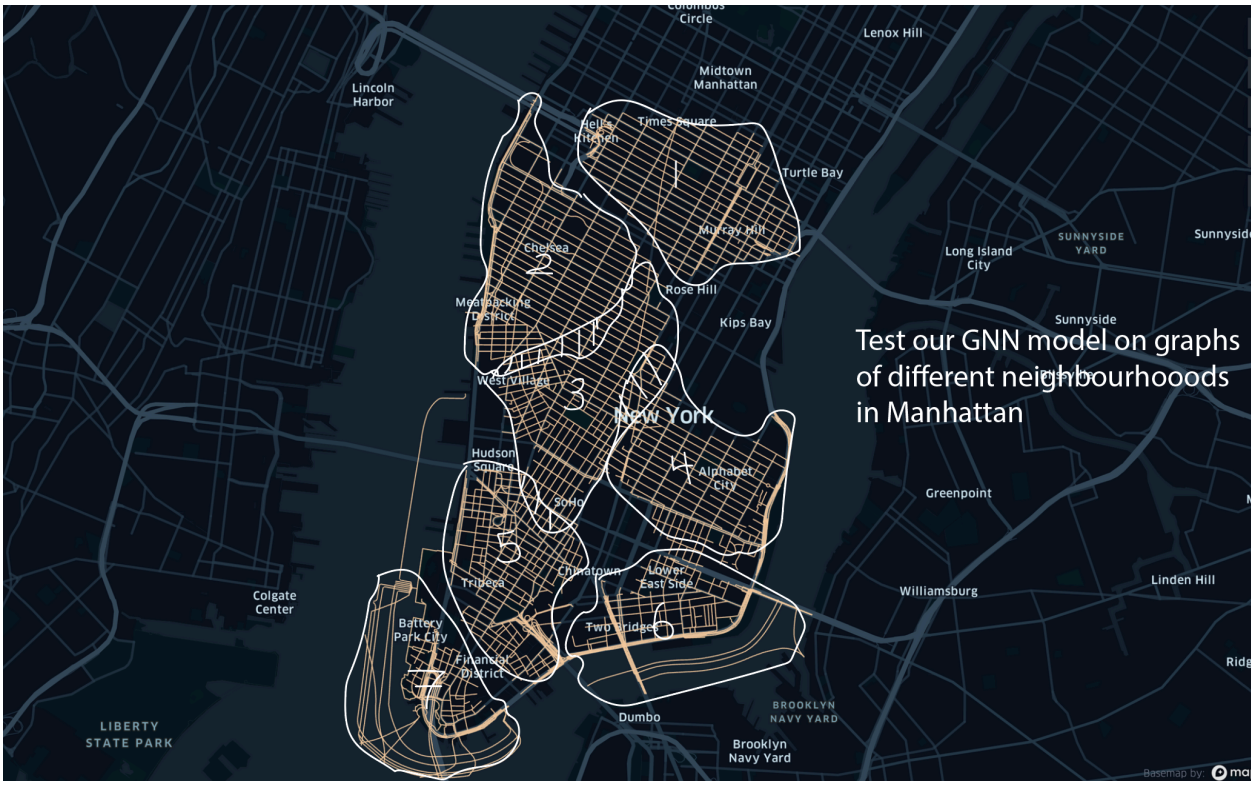
For the sake of clarity, it is useful to visualise the embeddings on the original map of New York. Here, we have visualised information from the 9 subgraphs of Manhattan in our test set. Each street has a 50 dimensional embedding space associated with it. We can interpret it as the probability that the street belongs to one of 50 classes, associated with street value. As can be seen below, the highlighted street has a 99.61% chance of belonging to class '1' of street value. There are some cases in which there is a much more split attribution of the street into classes. The Embeddings have been constructed, via judicious choice of the hot encodings, to be orthogonal to one another. That being said, class 0's interpretation (as houses on the street being worth $0-200/square foot) is far closer to class 1's interpretation (as houses on the street being worth $200-400/square foot) than class 50's interpretation (as houses being worth in excess of $2000/square foot). In cases where the street receives a high probability of two non neighbouring classes, we may assume that the street has a large mix of expensive and non-expensive dwellings.

It is legitimate to ask why we did not simply construct the model to perform a regression, as opposed to a classification. This is potentially a more fruitful route to take, and is seen as a

weakness of our approach.

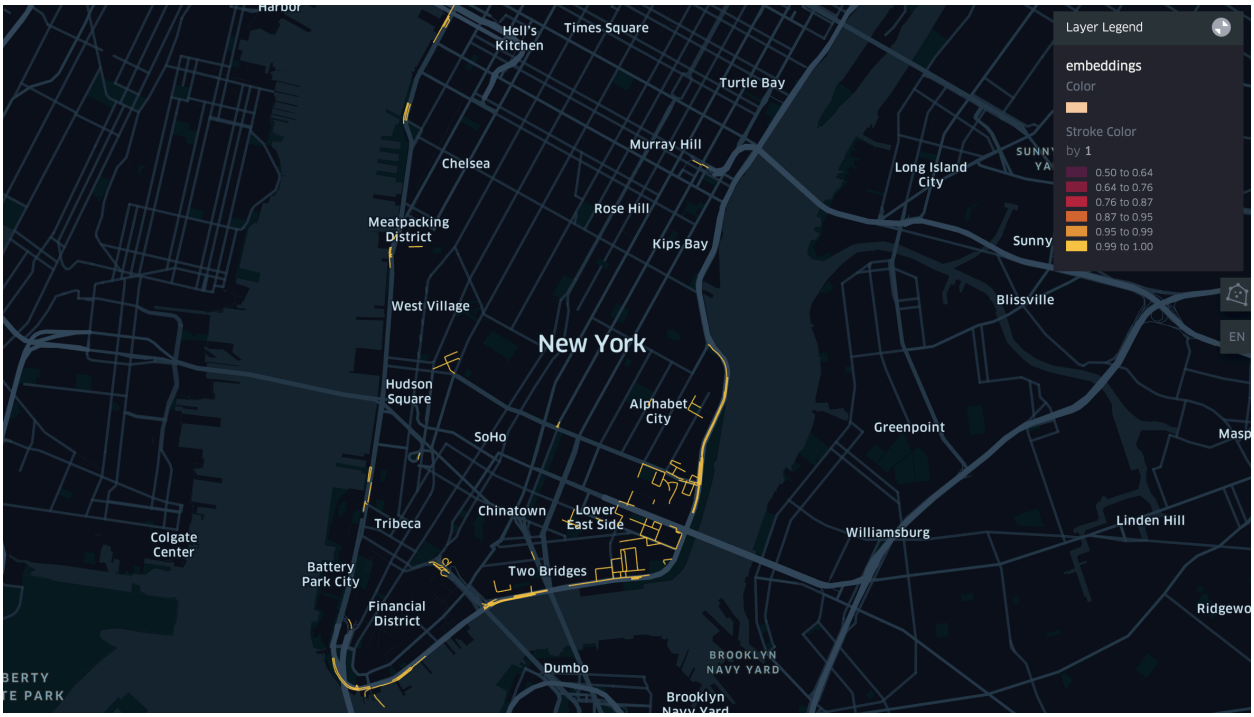


Coordinates in embedding space for one street visualised



To test the efficacy of the GNN at predicting house prices, we test it on a selection of different neighbourhoods in Manhattan.

We can visualise the values the projection of each street's projection space into one constituent basis. Here we are seeing the the values of the second dimension of each street's embedding space. This can be interpreted as the probability that houses on a street sell for between $200-400 dollars per square foot.



We can filter the results for greater clarity. above shows only those streets having a greater than 50% probability of having residential properties valued at $200-400 dollars per square foot. This shows what the model views some of the cheapest streets in New York It has largely picked sites that are coastal transit road, which in fact have almost no residential houses on them.

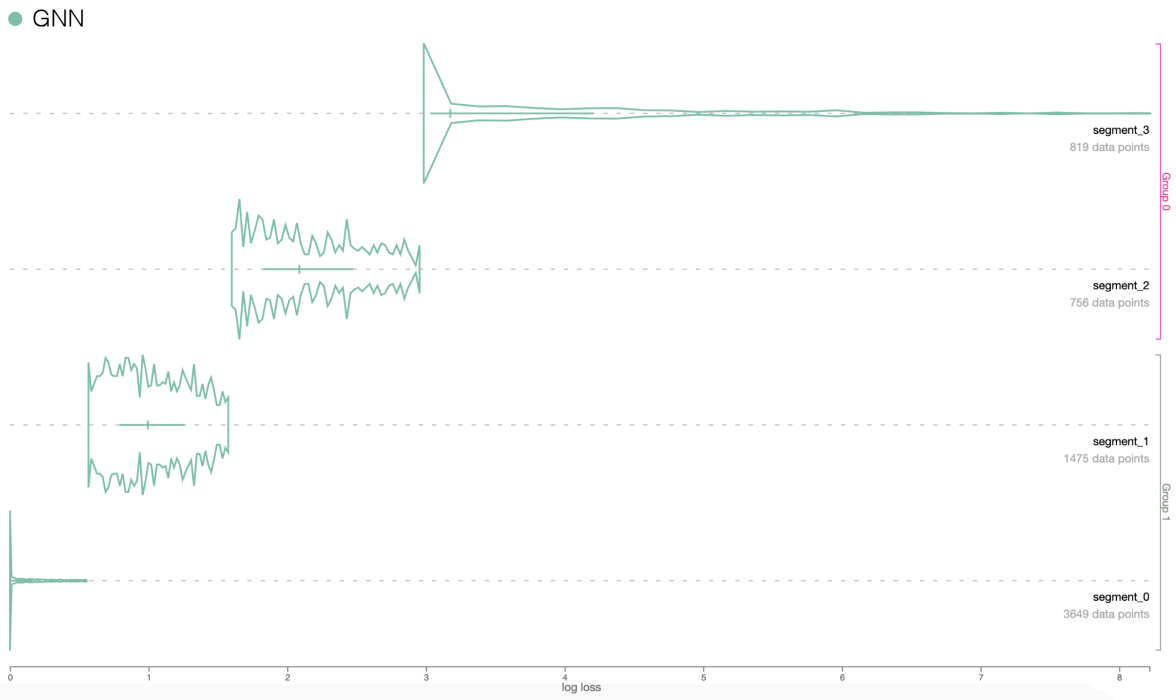
We can perform a multivariate analysis: all the highlighted streets have a greater than 50% chance of bering worth $600-800 /square foot. We can view the probability that a house on these streets are worth $200-400. Natural phenomenon in cities to have houses of divergent value on the same street

Feature analysis

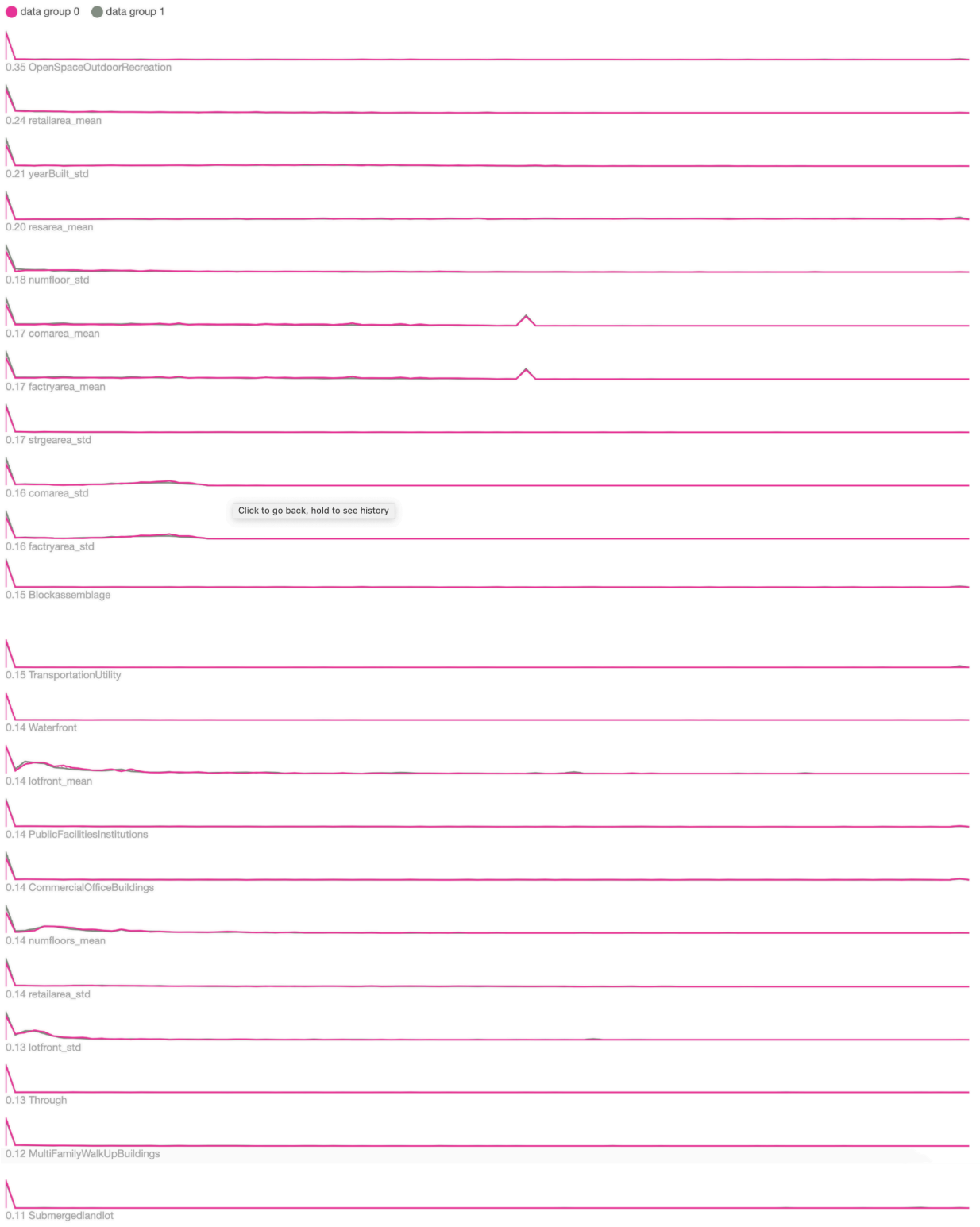
At this point, is is necessary to try and comprehend how the model has weighted various features as of importance in ascertaining house price. Equally, to see which features are more definitive predictors of the result, with less margin of error.

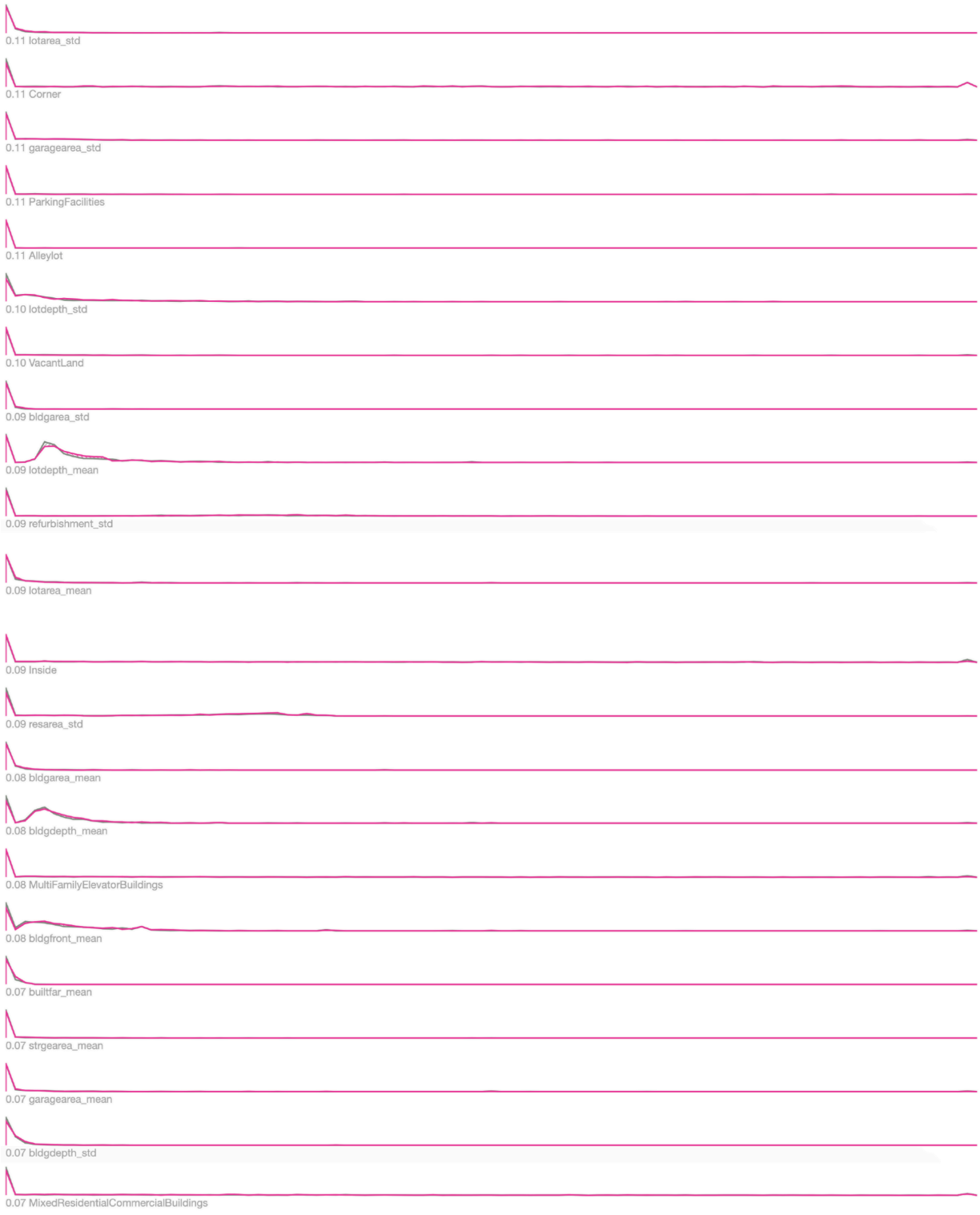
We would like to see whether there is a subset of data the model is inaccurately predicting, and what may be the cause of this. Below we have shown the feature distribution difference between better and worse-performing subsets of data. Below, 'Group 1' is the set of well performing data, whereas 'Group 0' is worse performing in the sense its ground truth is less accuratelt predicted by the model.

Data grouped by how well Model predicts its ground truth



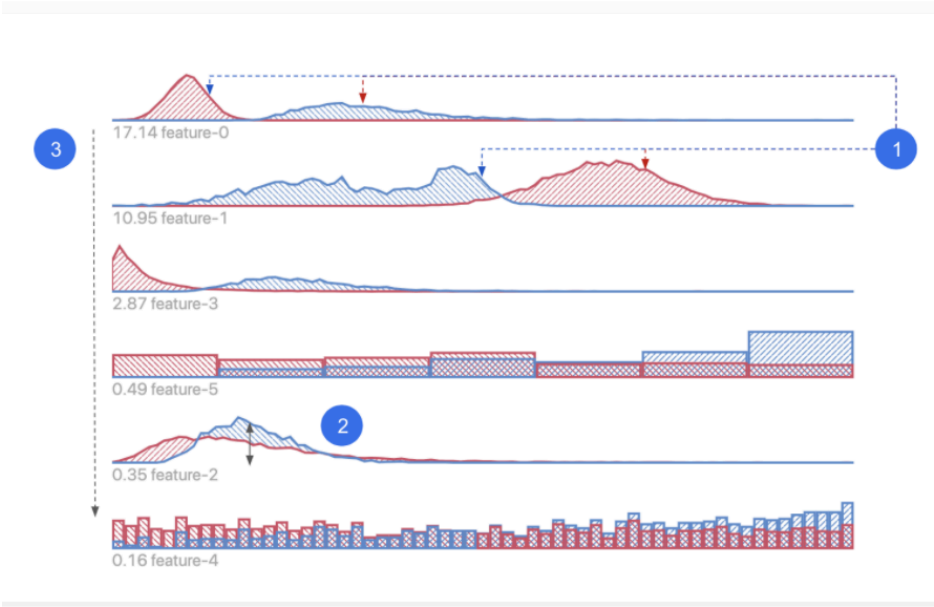
The x-axis represents the performance metric (log-loss), and the y-axis the data segments. Curve height shows performance distribution of the model on each data segment. Group 0 performs worse than **Group 1** at prediction.


data group 0
data group 1
0.35 OpenSpaceOutdoorRecreation
0.24 retailarea_mean
0.21 yearBuilt_std
0.20 resarea_mean
0.18 numfloor_std
0.17 comarea_mean
0.17 factryarea_mean
0.17 strgearea_std
0.16 comarea_std
Click to go back, hold to see history
0.16 factryarea_std
0.15 Blockassemblage
0.15 TransportationUtility
0.14 Waterfront
0.14 lotfront_mean
0.14 PublicFacilitiesInstitutions
0.14 CommercialOfficeBuildings
0.14 numfloors_mean
0.14 retailarea_std
0.13 lotfront_std
0.13 Through
0.12 MultiFamilyWalkUpBuildings
0.11 Submergedlandlot

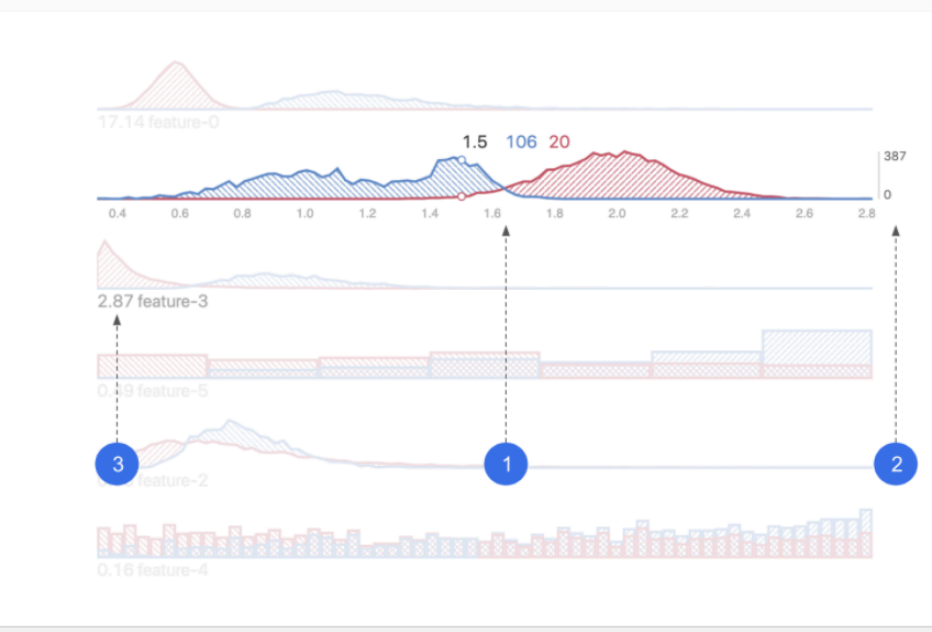

0.11 lotarea_std
0.11 Corner
0.11 garagearea_std
0.11 ParkingFacilities
0.11 Alleylot
0.10 lotdepth_std
0.10 VacantLand
0.09 bldgarea_std
0.09 lotdepth_mean
0.09 refurbishment_std
0.09 lotarea_mean
0.09 Inside
0.09 resarea_std
0.08 bldgarea_mean
0.08 bldgdepth_mean
0.08 MultiFamilyElevatorBuildings
0.08 bldgfront_mean
0.07 builtfar_mean
0.07 strgearea_mean
0.07 garagearea_mean
0.07 bldgdepth_std
0.07 MixedResidentialCommercialBuildings

Comparing the features sets from which the model well predicts (group 1) versus does not well predict (group 0) street values

▼ Some guidance on analysing the below charts [20]:



1. **Segment groups:** data slices you choose to compare against each other.
2. **Histogram / heatmap:** distribution of data from each data slice, shown in corresponding color.
3. **Ranking:** features are ranked by distribution difference between slices.



1. **X axis:** feature value.
2. **Y axis:** Data count/density.
3. **Divergence score:** measure of difference in distributions between slices

The clarity of the the graphs are hindered by the fact that the most common value (by far) for most features is zero. This squeezes other sections of the graph, making it hard to read.

for the feature (above), 1532 data points from group 0, and 5021 data points from group 1 have a value of 0. There are between 10-100 data points for other values between 0 and 0.5. We see,, for instance, that there at 19 and 77 data points with value 0.14. This makes it difficulty to view the discrepancy in features' values

This is an issue with the data collection, and cleaning.

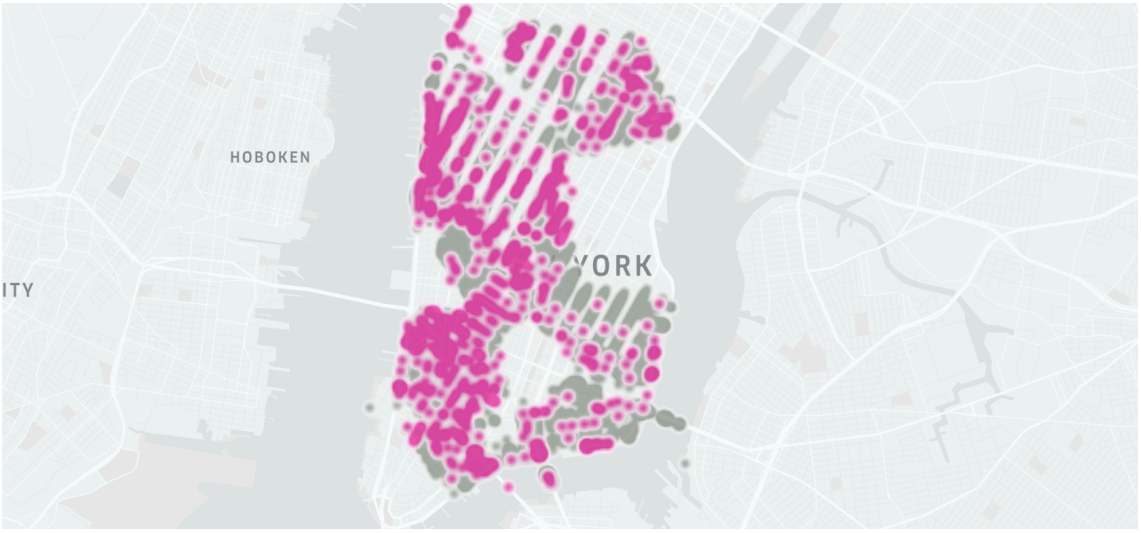
For features of the well-predicted, and no so well predicted models have virtually the same distrubutions. By looking at the few distrubutions which have discrpenacy, we can therefore pinpoint features which have a large impact on the proper identification of house price. These are:

- mean depth of the buildings on the street
- the mean number of floors of buildings on the street
- the standard deviation of the lot fronts

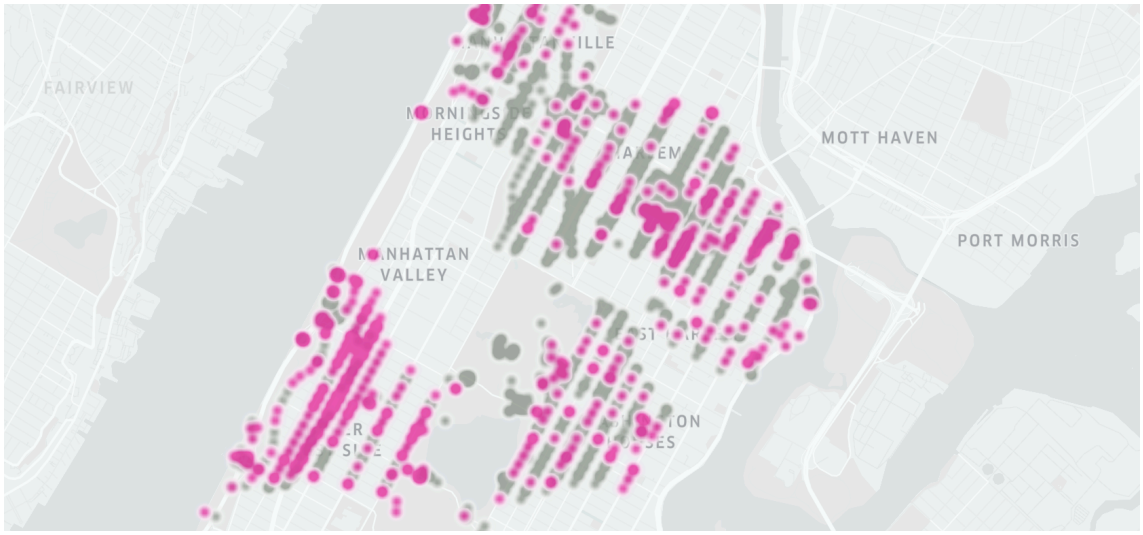
It is somewhat challenging to give any insightful information as to why this is the case, when there is probably still a large amount of noise in the model due to imperfect cleaning of the underlying data.

Comparing the locations of well predicted (group 1) versus not well predicted (group 0) street values by the model

We can also look back at the areas which were poorly predicted, and try and cast some light on why this was the case. As before, the **grey** (group 1) were predicted well for, and the **pink** (group 0) were predicted less well for.



For the neighbourhood clusters in the south of Manhattan, the model seems to predict especially well for the south-east, and less well for the south-west. There is no glaringly obvious discrepancies in performance.



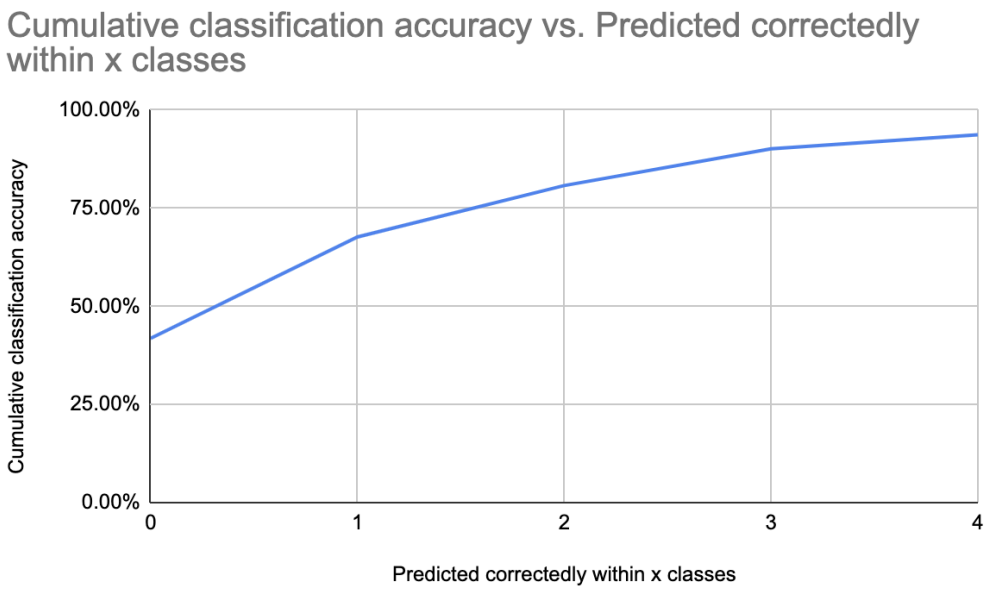
In the north of Manhattan, Harlem seems to be predicted particularly well for. The model performs more poorly on the Upper West side.

Discussion

Benchmarking

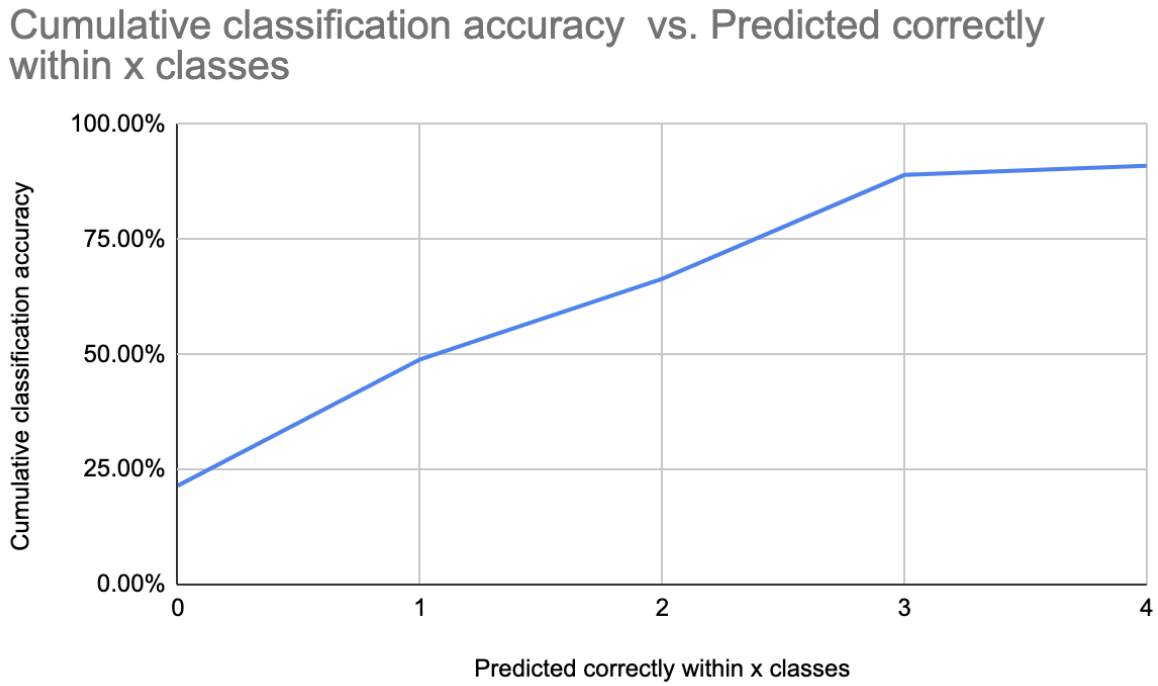
Although the results seem encouraging, we must benchmark them against more standard methodologies — there is little use using novel technique simply for their novelty alone

Attempt to understand the value of house prices is an old endeavour. The Boston housing dataset is regularly used as a primer into Machine learning methodologies. The GCNN network had been able to correctly classify 41.8% of streets into the correct category, 67.6% within one category of error, and 80.7% within two. Considering there are 50 categories, these results are encouraging.



Accuracy of GNN model

A standard deep neural network (of many layered perceptrons) can has some success in classifying this type of data, but with a considerable margin less accuracy:



Conclusion

The original question was: can Graph Neural Networks (GNNs) outperform Many Layered Perceptrons (MLPs) in ascertaining the value of properties on a given street. The results appear to suggest that GNNs are indeed better adapted to the task. Additionally, we can encode information GNNs in a way that is simply not possible with Many Layered Perceptrons. Integrating a notion of 'neighbourhood' or proximity in a MLP must painstakingly be encoded via backdoor approaches. In GNNs, the implementation is natural. It is almost certainly the case that we have yet to really explore the difference to its full potential — which may well highlight an even greater differentiation in performance. Such is for future work.

References

Many thanks to Dr. Sean Hanna for his incredibly balanced and thoughtful help.

[1]

Hillier, 2007

Studying cities to learn about minds: how geometric intuitions shape urban space and make it work.” Environment and Planning B: Planning and
Design;

Hillier, 2007

Space is the machine, A configurational theory of architecture

[2]

Turner and Penn, 2002

Encoding natural movement as an agent-based system: an investigation into human pedestrian behaviour in the built environment. Environment and Planning B: Planning and Design 29, 2002

Shannon and Weaver, 1949

The Mathematical Theory of Communication. Illinois University Press: Introduction.

[3]

Ostwald, 2011

The Mathematics of Spatial Configuration: Revisiting, Revising and Critiquing Justified Plan Graph Theory, Nexus Network Journal – Vol. 13, No. 2, 2011

[4]

Wu et al., 2019

A Comprehensive Survey on Graph Neural Networks, Journal of Latex Class files, VOL. XX, NO. XX, 2019 (https://arxiv.org/pdf/1901.00596.pdf)

Zhang et al.,

Graph convolutional networks: a comprehensive review, Computational Social Networks 6, 2019

[5]

Chaillou, 2020,

Space Layouts & GANs, (https://medium.com/spacemaker-research-blog/space-layouts-gans-2329c8f85fe8)

[6]

Tinghua, 2020,

A graph convolutional neural network for classification of building patterns using spatial vector data

[7]

Wiltschko et al.,

Machine Learning for Scent: Learning Generalizable Perceptual Representations of Small Molecules, 2019 (https://arxiv.org/abs/1910.10685)

Harada et al.,

Dual graph convolutional neural network for predicting chemical networks, BMC Bioinformatics volume 21, Article number: 94, 2020

[8]

Defferrard et al.,

Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering, Neural Information Processing Systems (NIPS), 2016.

[9]

Allamanis et al., 2018

Miltiadis Allamanis, Marc Brockschmidt, and Mahmoud Khademi. Learning to Represent Programs with Graphs. In International Conference on Learning Representations (ICLR), 2018. (https://arxiv.org/pdf/1711.00740.pdf)

Brockschmidt, 2019

Marc Brockschmidt. GNN-FiLM: Graph Neural Networks with Feature-wise Linear Modulation. (https://arxiv.org/abs/1906.12192)

Li et al., 2015

Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated Graph Sequence Neural Networks. In International Conference on Learning Representations (ICLR), 2016. (https://arxiv.org/pdf/1511.05493.pdf)

Ramakrishnan et al., 2014

Raghunathan Ramakrishnan, Pavlo O. Dral, Matthias Rupp, and O. Anatole Von Lilienfeld. Quantum Chemistry Structures and Properties of 134 Kilo Molecules. Scientific Data, 1, 2014. (https://www.nature.com/articles/sdata201422/)

Schlichtkrull et al., 2017

Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling Relational Data with Graph Convolutional Networks. In Extended Semantic Web Conference (ESWC), 2018. (https://arxiv.org/pdf/1703.06103.pdf)

Sen et al., 2008

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective Classification in Network Data. AI magazine, 29, 2008. (https://www.aaai.org/ojs/index.php/aimagazine/article/view/2157)

Veličković et al. 2018

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. In International Conference on Learning Representations (ICLR), 2018. (https://arxiv.org/pdf/1710.10903.pdf)

Xu et al. 2019

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How Powerful are Graph Neural Networks? In International Conference on Learning Representations (ICLR), 2019.

(https://arxiv.org/pdf/1810.00826.pdf)

Zitnik & Leskovec, 2017

Marinka Zitnik and Jure Leskovec. Predicting Multicellular Function Through Multi-layer Tissue Networks. Bioinformatics, 33, 2017. (https://arxiv.org/abs/1707.04638)

[10]

Gilmer et al., 2017

Neural Message Passing for Quantum Chemistry, 2017 (https://arxiv.org/pdf/1704.01212.pdf)

[11]

Law et al., 2019

Take a Look Around: Using Street View and Satellite Images to Estimate House Prices, 2019 (https://arxiv.org/abs/1807.07155)

[12]

Wu et al., 2019

A Comprehensive Survey on Graph Neural Networks, Journal of Latex Class files, VOL. XX, NO. XX, 2019 (https://arxiv.org/pdf/1901.00596.pdf)

[13]

Otness, 2019

Graph Convolutions and Machine Learning, http://nrs.harvard.edu/urn-3:HUL.InstRepos:38811540.

Bronstein, 2020

Learning on non-Euclidean domains, Lecture 11 (https://vistalab-technion.github.io/cs236781/lecture_notes/lecture_11/)

[14]

Wu et al., 2019

Simplifying Graph Convolutional Networks, 2019 (https://arxiv.org/abs/1902.07153)

[15]

Bruna et al., 2013

Spectral Networks and Locally Connected Networks on Graphs, 2013 (https://arxiv.org/abs/1312.6203)

[16]

Hammond et al., 2011

“Wavelets on graphs via spectral graph theory,” Applied and Computational Harmonic Analysis, vol. 30, no. 2, pp. 129–150, 2011.

[17]

Kipf, 2020

Deep Learning with Graph-Structured Representations, 2020 (https://hdl.handle.net/11245.1/1b63b965-24c4-4bcd-aabb-b849056fa76d)

Kipf, et al. 2017

Semi-Supervised Classification with Graph Convolutional Networks, 2017 (https://arxiv.org/abs/1609.02907)

[18]

Data AI group at Microsoft Research, Cambridge, UK,

Graph Neural Networks in TF2, (https://github.com/microsoft/tf2-gnn)

[19]

Bui and White, 2019

Mapping the Shadows of New York City: Every Building, Every Block, New York Times 2019

[20]

Uber Manifold,

https://github.com/uber/manifold

https://hectorcrean.typeform.com/to/ujifV9