is an order to place the nodes within a graph[2]. The key to understanding a graph structure is to incorporate neighbor information. Linear propagation builds a weight for each first order neighbor node [3,4]. Directed edges identify which nodes need to aggregate neighbor weight information. Message Passing Neural Network is the formal method for aggregating information between nodes until all nodes are updated [5]. The learning functions work better by normalizing the data, but the algorithm does not manage self-loops [6]. Graph convolution network (GCN) adds a self-loop weight to the algorithm and renormalization of current node and its neighbors' data [7]. The final encoding is a sum of the current node and neighbors' data all averaged together for a final value. Graph SAGE improves GCN by replacing the symmetric normalization with a random walk normalization [8]. In a complex graph structure, first order neighbors may not give enough information to encode a graph structure, though high-order neighbor algorithms look to fix that problem [9]. Researchers try to update neighbor information by using a convolutional neural network [10]. In these networks, Node2vec turns every node into a one-dimensional vector, then combines neighbor and target vectors together as a node update method [11].The focus of high-order node algorithms is to avoid over-smoothing. This is because adding more neighbors to a node can smooth out local information. High order algorithms are useful in representing complex signals when a flexible order is possible with proper constraint [12]. Many GNN algorithms focused on the forward propagation of labels or features within a graph structure. Rational propagation also focused on the reverse direction propagation to mitigate over-smoothing in graph structures [13]. Convolutional neural network and Page rankwere applied to GNN to propagate information within a graph network [14]. The concept applies a rational function to the node's adjacency matrix. The domain of spectral GNN is to study the relationships between polynomials, eigenvalues, and eigenvectors of the graph matrix [15]. Since the main idea of this study is to view a supply chain as a graph, it doesn't include spectral information. For large scale graph structures, sampling method is a useful technique and improves computation time of the analysis [16].

In literature, convolutional neural network models were applied for modeling structured sequence of the data [16]. Their model compared the graph vectors for similarities and then grouped them in different clusters. In GC_LSTM, graph convolution embedded LSTM for dynamic link prediction [17]. In this article, the authors' model uses a LSTM to observe the one-dimensional vectors for patterns to predict which nodes should have links. Link prediction can be applied as a way to understand how entities are connected in a supply chain since it may not be apparent [2]. Transfer graph neural networks can be also applied for nodal level predictions such as pandemic forecasting in which Message Passing Neural Network (MPNN) encoded the graph structure and LSTM learnt how the virus spread to different regions [18]. The study [25] uses a GAT-LSTM pipeline to understand airport throughput. This study first believed relations would improve GAT, but [26] shows that this is not the case. Another study [28] uses cardinality to augment GAT, but cardinality does not represent influence.

Other than neural networks, other data analysis techniques were applied to the network structure. For example, prophet and Auto-Regressive Integrated Moving Average(ARIMA) were applied for graph modeling [21, 22]. A Bayesian network can also measure the influence of one node on another node within the network. The influence being calculated is a whole network influence for a node. Information theory helps to propagate that influence through the network [31,32]. This study uses the Bayesian influence value to see what percentage of failure propagates from a node. Artificial intelligence can be coupled with a supply chain simulation engine as in [33]. The supply chain simulation engine supplies the required data for an artificial intelligence to predict future supply chain events. This study uses a similar technique focusing on risk management. The authors in [34] studied a variety of deep learning models for demand forecasting

showing the benefit of deep learning over transitional methods. For the same goal, the [35] uses data mining and business intelligence methods to pull key points from a supply chain dataset. In [36] simple multi-layer perception neural networks are used to investigate supply chain risk management, but their proposed technique does not include network structure features and node communications in the analysis. This makes the predictions vulnerable to any potential disruptions that may happen in part of the supply chain. Data mining supply chain using GNNis able to discover many supply chain items helping a company identify risk area as shown in [37]. Although this research proposed using GNN in supply chains analytics, ignoring time-series prediction of risk values and complexity of the technique hinders its practical application in highly volatile markets.

To fill the existing gap in the literature, this study proposes a novel approach to evaluate node level risk metrics in supply chain. This is implemented by augmenting graph attention networks (GAT) and replacing starting attention values with centrality measures. The GAT encodes a graph to be used by a LSTM network for predicting future values. Output from the GAT+LSTM is processed by a risk analysis algorithm. This research contributes to the literature on the following aspects:

- Proposing a new metric for risk evaluation of supply chain networks while considering mutual effects of the nodes on each other and final outcomes of the business.
- Integrating centrality concepts with GAT networks to increase its efficiency and computation time for time-series predictions.

The rest of the article is organized as follows. Section 2 covers the methods and is broken into four subsections, section 2.1 is the dataset structure and creation method, section 2.2 is the graph data structure, section2.3 details the graph neural networks, and section 2.4 details the risk equation. Section 3 details the experiments and results and, finally, section 4 presents the conclusion and directions for future research.

METHODS

The concept of relating a supply chain to a network of entities allows the research to use state-of-the-art tools like graph neural networks. GNNs are a specialized type of deep learning models that work on graph data. GNN can understand the structure, dependances, and position of a node within a graph whereas other neural networks cannot view graph data because the positioning of nodes is not apparent. The closest related neural networks to GNN are visual neural networks which view their data as a grid. However, the problem with grid data is that it is dense compared to graphs that are sparse. Namely, grids do not have a direction dictated by their relationship to other nodes within their structure. These are merely a couple of reasons for using GNN and not standard deep learning models. This study demonstrates that GNN can understand highly dependent data [1], the positional structure of a supply chain, and the flow or direction of products through a supply chain [2]. Using a message passing neural network on a supply chain network works very well for encoding, since a supply chain network holds dependent data. Message passing neural networks are good at capturing the dependence and positional nature of data, and therefore this research uses data that is time dependent. Long Short Term Memory (LSTMs) are useful in understanding the dependance within a time series, which is the type of data found within a supply chain. This research, similar to previous work in other applications, assumes aMPNN+LSTM model is able to understand data dependances and data spreading over-time.

This research applied fast graph representation learning with PyTorch Geometric which is a python library that

allows the research to build, train, validate, and evaluate quickly without having to hand build basic neural networks [29]. It also contains MPNN and LSTM models and has tools for the in-memory graph structure and the batching process for faster learning. In the current study, the graph temporal library entitled Geometric Temporal: Spatiotemporal Signal Processing with Neural Machine Learning Models [30] which is built on top of PyTorch Geometric.

This study starts by creating a supply chain dataset by running an algorithm that populates a supply chain network with capacity data and order data. The dataset is read into memory for processing by different types of GNN. The GNN predicts future values for capacity at each supply chain node. This study combines the GNN output with ARIMA output, network centrality values, and Bayesian influence values to create a supply chain risk management value. The risk management value gives this study a node's risk of failure.

Dataset

The current research collects values from several data sources [19] [20] to create a single supply chain network. The supply chain dataset compilation contains a network of suppliers, manufacturers, distributors, and customers, and treats them as supply chain entities. Supply chain entities are a generic data type that the graph neural network uses for processing information. Supply chain entities have one feature, or capacity, as a data item. Capacity data from a table that holds supplier data type capacity within the columns and the rows represent a day. The other nodes' capacities are generated by averaging incoming capacities. A variance variable modifies each node, except suppliers, to add realistic issues to the supply chain like quality problems, delays, etc. The current research collects order relationship data between supply chain entities from a table. The table has time, in node, and out node data.

The research treats supply chain entities as nodes within a supply chain network, and therefore, an out node ships the products to an in node. The capacity data for supply chain entities other than suppliers is entirely dependent, and supplier capacity is the only non-dependent data. Manufacturers, distributors, and customers' capacity data is an average of incoming capacity from entities higher up the chain. This study uses these two data items for its GNN algorithm, and each edge has a weight showing items order by a node.

Graph Data Structure

The algorithm reads in the data from a dataset file, detailing network structure and node values. Reading a row of data creates one graph for a time snapshot. Using the Network X python library, a directed graph structure was created for this study, and uses the out node and in node fields as labels for node within the supply chain graph [37]. Adding an edge to the supply chain graph creates two nodes and one edge, and notably, the direction of the edge is from out node to in node. The quantity ordered is the weight value for the edge. This research stores a normalized weight value in the graph structure.

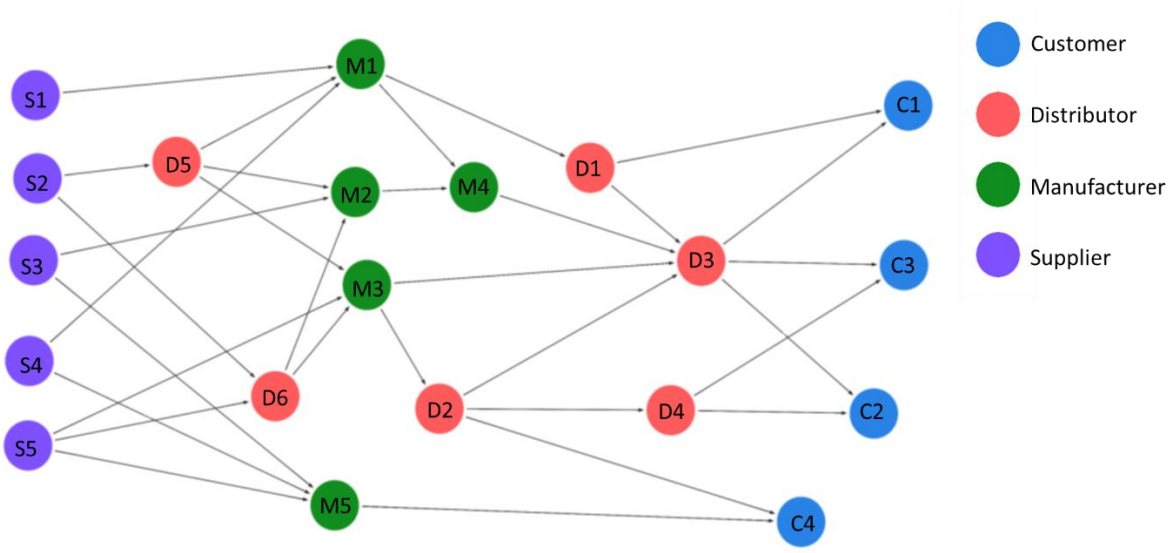

Figure 1: The Default Supply Chain Network.

Reading a row creates all the nodes and edges for the supply chain graph. This study's supply chain graph has 20 nodes and 32 edges. The graph represents the supply chain at a time snapshot. Reading each row in input file creates 360-time graph snapshots. The graph dataset contains 360 directed graphs without node features after the first processing step. The research injects supplier node features, or capacity value, from an input file into the graph. Reading each row in an input file, the supplier node can be found at the correct time snapshot and add to its capacity value. However, this study only applies normalized capacity value to each node. A simulation algorithm walks through the graph network, generating capacity values for other nodes. The values generated for manufacturers, distributors, and customers are an average of the incoming capacity values. The simulation algorithm adds a variance parameter to each node type. Manufacturers have a quality variance, and distributors have a delay variance, and variance does not affect customers and suppliers. The research now has 360 graph snapshots, with each graph holding capacity value as node features, items ordered as weighted edges, and edge direction. After processing all of the input, the graph structure for the GNN algorithm becomes apparent. The simulation method gives a time snapshot of a graph in the dataset, and the graphs are in a stack starting from the first-time entry. In this study, access to a graph in the stack is ordered by time. Restricting the order of access allows this researcher to disregard the time column in the dataset. The GNN is observing 7,200 data points in two dimensions. The first dimension is across a single graph, second dimension is across time.

Graph Neural Networks

Graph neural networks are a form of deep learning based on graph structured data. The artificial neural network learns the structure and data held within the graph. Learning can help with tasks such as classification, clustering, link prediction, and value prediction. The reason for using a GNN is that standard deep learning models have difficulty understanding dependent data. Dependent data relates to many real-world problems, such as social networks, network analysis, and supply chain. A neural network starts by assembling an adjacence matrix to represent a graph data structure. Each edge is assigned a value of one when present, and zero is assigned if there is no edge. Columns and rows are nodes, and data within the matrix is an edge. This research uses a directed graph, so an edge from node 1 to node 2 would show a 1 in the matrix, but node 2 to 1 would be a 0. The diagonal in a matrix represents a self-loop on a node.

In the next part of the graph, the features and parameters of each node are to be embedded into the data input.

Again, we can represent the features as a 1-dimensional vector held in a matrix. The degree matrix displays the degree of each node down the diagonal. Subtracting the degree matrix by the adjacence matrix creates a Laplacian matrix. Multiplying the Laplacian matrix by the feature matrix will create a new 1-dimensional feature vector for a node within the graph. However, the problem with this method is the feature vector is shallow and does not take neighbors into count. The reason shallow vectors are a problem is that graphs have an arbitrary order, so using a simple embedding may not represent the actual graph, but rather a variant graph. Updating each node's feature vector with its neighbor's feature vector will give the data local information. An embedding created from a node plus its neighbors feature data will better represent the data, location, and dependent data. The model can use each one of the feature vectors as input for its neural network algorithm for classification, prediction, etc. There are different methods for updating node data with neighbor data. The present research uses the message passing method, and a modified graph attention algorithm.

The GNN model can use a message passing concept to update a node's hidden state, or feature vector, with neighbor's hidden states. The node being updated will receive messages from its neighbors, m_t. A function, m, will aggregate those hidden states, h, and edges weights, e, between node v and w at time step t(3). The aggregate function can be a simple summation or convolution (4), either creates a single message.

$$m_v^{t+1} = \sum_{w \in N(v)} M_t(h_v^t, h_w^t, e_{vw}) \quad (3)$$

Using that message, the node's hidden state updates using the following Equation (4). U_t is an update function for that model.

$$h_v^{t+1} = U_t(h_v^t, m_v^{t+1}) \quad (4)$$

Every node uses the above equations to update its hidden state at time step t. Updating the hidden states will happen for a set number of time steps, t. After all the time steps have processed, a single graph, G, feature vector is readout, $\hat{y}$, using Equation(5).

$$\hat{y} = R(\{h_v^t | v \in G\}) \quad (5)$$

A common way to build the aggregation function and combine or update function is to use a convolutional neural network (6), where D, is an identity matrix, A is an adjacency matrix, W is a weighted matrix, H is a convolutional layer, and k is the iteration. Equation (7) shows how to combine the values together.

$$H^{k+1} = \sigma(D^{-\frac{1}{2}} A D^{-\frac{1}{2}} H^k W^k \quad (6)$$

$$H_i^k = \sigma(\sum_{j \in (i)} \frac{A_{ij}}{\sqrt{D_{ii} D_{jj}}} H_j^{k-1} w^k + \frac{1}{D_i} H_i^{k-1} W^k) \quad (7)$$

An MPNN using convolutional aggregation as an update message function is an efficient method when all the neighbor nodes are of equal importance. Learning the importance of a neighbor node by using a GAT as the aggregation function instead of convolutional aggregation introduces the concept of influence [23]. A GAT uses an attention parameter as an importance weight, W, for a node h. The model learns the importance weight during a training session. Equation (8) calculates the attention parameter, a. In Equation (8) a is transposed and Leaky ReLU is an activation function and || means to concatenate the values.

$$e(h_i, h_j) = LeakyReLU(a^T \cdot [Wh_i || Wh_j]) \quad (8)$$

Then the values are normalized using a Softmax function in Equation 9.Nonlinearity σ is applied to the summation of N, neighbor nodes in (10).

$$a_{ij} = softmax_j\left(e\left(h_i, h_j\right)\right) = \frac{\exp(e\left(h_i, h_j\right))}{\sum_{j' \in N_i} \exp(e\left(h_i, h_{j'}\right))} \qquad (9)$$

$$h_i^{'} = \sigma(\sum_{j \in N_i} a_{ij} \cdot Wh_{j)} \qquad (10)$$

Equations (8), (9), and (10) make up the GAT algorithm. This research uses a weighted graph to introduce the concept of importance by centrality. In this study, four different centrality measures were tested as the weight values for each edge in the graph. Furthermore, the present work applies a centrality measure to the whole graph, meaning the edge weight represents one of the following: degree, betweenness, Eigen values, or Katz values' centrality. Degree centrality is the sum of in and out vectors for a node normalized whereas betweenness is a rating based on the shortest path algorithm. The more times a node appears on a short path, the higher its rating. The betweenness Equation (1) for node v where node s is the start of the path and node t is the end, σ_{st} is all the shortest paths, $\sigma_{st}(v)$ is shortest path with v, not as the end node.

$$\boldsymbol{g(v)} = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}} \quad (1)$$

Eigen (2) values are the eigen vector for a node, which represents the node's importance to its neighbors. In Equation (2) A is the adjacency matrix or λ a constant and x is a one-dimensional matrix of node degrees. Katz values are a more generalized version of the Eigen values.

$$Ax = \lambda x \quad (2)$$

This study's GNN algorithm learns the structure, dependence between nodes, and time periods so it can make predictions on node capacity for the future supply chain graphs.

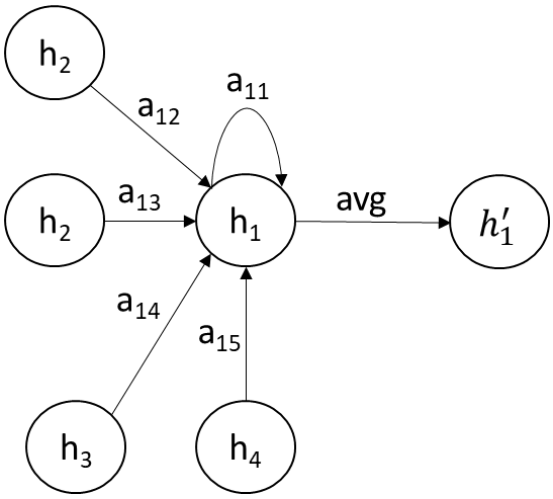

Figure 2: GAT with Single Head and Centrality Values.

Figure 2 shows a standard GAT where all the attention values are averaged together but, in this study, the starting a value are replaced with centrality values. Katz centrality gives a node's importance to its neighbors, so the research uses Katz as the weight values for the experiments. The GAT sums the edge weight, Wh_i, and Wh_j together to calculate the attention value (10). The research replaces the random starting weights in a GAT with centrality values to help in training

the model and accuracy.

A deep learning model can feed the readout into another model for further processing. In this study, the readout vector was fed into a LSTM model. The LSTM captures the temporal dependence between each graph in a time step. The LSTM will understand the trend that can lead to node values changing over time and is able do this better than other neural network models because it can remember important data. LSTM designs its neural network on the idea of memory cells, similar to memory cells used by a computer. The LSTM cell takes in the previous cell state, a hidden state, and the input. Using gates and functions, the cell learns what states to keep and merges that data with the input data to produce output data. Figure 3 shows the flow of data through the cell.

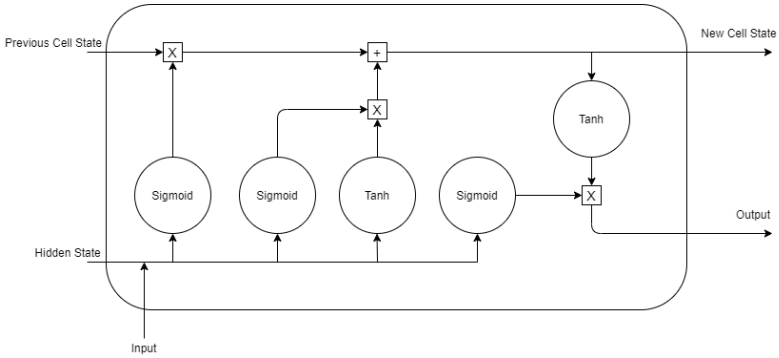

Figure 3: LSTM Model.

The LSTM cell takes in the hidden state of the node and the input. On the left side, the process begins and ends on the right. The top path creates the next cell state for the LSTM cell whereas the bottom path creates the output data for the LSTM cell. The first sigmoid function takes the input and hidden state and outputs a value between 0 and 1. The reason for using a sigmoid function is that it acts as a forget gate in the LSTM cell. Values of zero mean to forget, while values of one mean to remember. The value coming out of the first sigmoid function is multiplied by the previous cell state. As time goes on, the sigmoid function learns the importance of the previous cell states. The second sigmoid function rates the importance of the hidden state and input. The first tanh function helps to control the data and stabilize the value. The output of the second sigmoid is multiplied by the first tanh function to produce a value. The cell adds the value to the current cell state, creating the cell state for the LSTM. The third sigmoid function acts as an activation function, taking the hidden state and input, creating a value v_s. The second tanh function also takes the current cell state and creates a value v_t. The LSTM multiples v_s and v_t, giving the final output for the LSTM. In summary, the first three functions build the new state of the LSTM cell, while the last two functions build the output value thereof. The research uses an MPNNor GAT model to learn the structure and data dependencies of a supply chain network at specific time. The MPNN/GAT encodes the data in the supply chain network for consumption by the LSTM model. The LSTM takes each one of the encoded supply chain graphs as input. The LSTM links previous supply chain graphs to current supply chain graphs, so the high and low of the supply chain system can be identified in the current study. AGNN can understand any graph structure data by creating the data pipeline from the two neural network models. Figure 4 illustrates the pipeline GNNused to predict valuesthat are validated on previously observed data. Notably, the pipeline can be used to predict future values once trained.

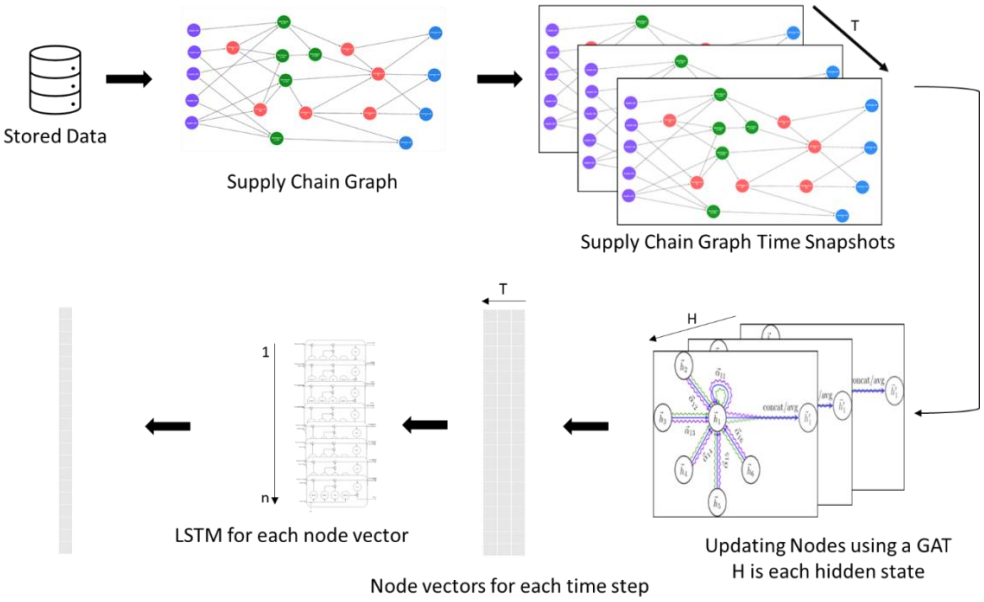

Figure 4:GNN Model Pipeline.

The reason for using MPNN+LSTM is that each supply chain graph is a three-dimensional network of entities, or a spatial graph [18].Each graph is a time snapshot that can stack one on top of one another. The model allows for an encoding layer to change, since MPNN is a simple concept. This study in specific replaces the MPNN with a GAT to improve accuracy, and to further improve accuracy, this research augments the attention values with Katz centrality values.

Graph neural networks' advantage is their ability to embed graph information and feature information into 1-dimensional vectors. Node feature vectors can combine into a single graph vector for other models. Classifying and clusters vectors are possible models, and predicting node states, links, and label is also possible. Comparing vectors to each other can show similarities. However, graph neural networks have limitations such as scaling issues as specific graph neural networks have difficulty scaling to larger graphs. Moreover, graph neural networks can be computationally expensive to run because there is not a single algorithm that can encode and decode every graph. Graph datasets require a neural network pipeline, or a linear chain of separate neural network models.

The influence of one supply chain entity on another is important to calculating risk in a supply chain. The amount of influence an ancestor has on a decedent is like the mutual information equation [31,32] in information theory (Equation 11). The equation shows the amount of uncertainty that a node removes from the target node. Nodes directly connect to one another remove the most uncertainty. Removal of uncertainty and influence are similar in this situation. Using the mutual information equation within a Bayesian network requires a network structure, node probabilities, and conditional probability tables.

$$MI(X,Y|Z) = \sum_{x,z} P(x,z) \sum_{y} P(y|x,z) \log_2 \frac{P(y|x,z)}{P(y|x)} \quad (11)$$

Knowing the state of node X, how much uncertainty does node X remove from node Y. If node X and Y are not adjacent to each other than nodes in between, need to be considered. Z is all the node states for the nodes in between node X and Y. These probabilities are part of Equation 11.

Risk Calculation

This study shows how the data from a GNN can be part of a supply chain risk calculation. It expands on the standard risk of node n at time t (R_{nt}) by breaking down probability (P)and loss (L) into separate node values, thereby creating a granularity to the standard risk formula that considers how each node affects the supply chain outcomes.

$$R_{nt} = P_{nt} * L_{nt} \quad (12)$$

$$P_{nt} = A_{nt} * B_{nt} \quad (13)$$

$$L_{nt} = M_{nt} * I_{nt} \quad (14)$$

In the above formulations, Equation (12, 13, 14)shows this study's concept for calculating risk by using the ARIMA function to predict future failure values for each node, A_{nt}, in the supply chain. Each node's historical data is input for separate ARIMA functions centering on a single node. For the purpose of this study, the betweenness value, B_{nt}, is calculated for each node based on the supply chain's network structure and is understood as the value of how important a node is to the structure of the supply chain and production flow in the system. The GNN considers the neighbor node's values when it is trained to learn how to predict future node values. The present research predicts future node values but only uses the end nodes, M_{nt}, since they represent the eventual outcome of the supply chain. A profit or loss outcome, I_{nt}, represents the node influence values. The influence of a node on another node is important to a supply chain. Knowing the influence value of a node on an end node helps to predict the impact of losing that node. A Bayesian Network can calculate influence values because of its distributed probability structure. Combining these values together, this study creates a per node risk value that considers future values, importance, and node future outcomes, and a node's influence on end node values. Considering all these values within a risk calculation allows researchers to build a quality risk metric for future decision-making tasks.

RESULTS

This study's graph neural network model uses python, PyTorch geometric, and PyTorch geometric temporal libraries to train, evaluate, and validate algorithm performance. The Network X python library defines the in-memory graph data structure for use by the graph neural network algorithm. The model is trained using the data collected for T days and it attempts to predict remaining capacity values for a supply chain graph afterT+1 day. The model selects the best MPNN/GAT model to use in the next step and passes the data through two MPNNs/GATs and two LSTMs, and finally the output data has a linear transformation apply to smooth out the data. The model evaluates final output predication against the ground true dataset using the follow formula to calculate mean error (13). Where $\hat{y}$ is output of the GNN, y is validation values, t is time, v is set of output values.

$$error = \frac{1}{ndt}\sum_{t=T+1}^{T+dt} \sum_{v\in V} \sum_{v\in V} (\hat{y}_v^{(t)} - y_v^{(t)}) \quad (15)$$

The above experiments are running on an AMD Ryzen 9 with 8 cores, 16GB of ram, and Nvidia RTX 2060 GPU. Results for the Graph Neural Network models take about 16 minutes to return. The research compares the MPNN+LSTM model to the Prophet and ARIMA algorithms which are autoregressive models with a moving average [21,22]. As a baseline, this study uses the average capacity (AVG) up to the day of testing. The second baseline used in this research is the average window (AVG_WINDOW), where the average capacity for the past days and day, d, is the window size. Table 1summarizesthe results of basic time series algorithms against an MPNN GNN and other neural networks such as GAT

with Centrality + LSTM model. Mean squared error (MSE) is this study's comparable value and is defined as an error between the predicted value and the actual value in the validation set.

Table 1 shows that Prophet and ARIMA have large mean squared errors, greater than 1. The two baselines, AVG_WINDOW and AVG, show a mean squared error above 1. Table 1 shows that Prophet and ARIMA are not much better than AVG and AVG_WINDOW. The mean squared error drops dramatically with the MPNN model. The MPNN model's error is below 1. Table 1shows the model using GAT and LSTM to evaluate the data. GAT with centrality + LSTM further increases accuracy, as Table 1 shows. GAT with centrality + LSTM has the best accuracy values, hence proving centrality can help GAT achieve accuracy improvements. As the results show, GAT with centrality + LSTM improves accurate predictions of capacity within the model which is mainly because of its ability in mapping internal connections and dependencies inside the supply chain network.

Table 1: Mean Square Error of Different Prediction Models

Model	Mean_Error
PROPHET	2.31457
ARIMA	1.85811
AVG_WINDOW	2.35932
AVG	2.21412
MPNN	0.38272
GConvGRU	0.0796
GCONVLSTM	0.0827
MPNN+LSTM	0.0796
GAT+LSTM	0.0721
GAT+LSTM	0.08
GAT with Centrality + LSTM	0.0712

Design of experiments based on multi-variate analysis of variance (MANOVA) are set to find the best settings of hyper parameters for the model to reduce computation time and MSE (Figure 5).For this study, the hyper-parameters for MPNN+LSTM, GAT+LSTM, and GAT with centrality + LSTM were set by modifying learning rate, hidden layers, and dropout. The experiment trains each model for 1000 epochs and uses a batch setting of 8 graphs. Batching allows for PyTorch geometric to work on graphs in parallel instead of one by one [29]. This study ran 16 experiments using different hyper parameters.

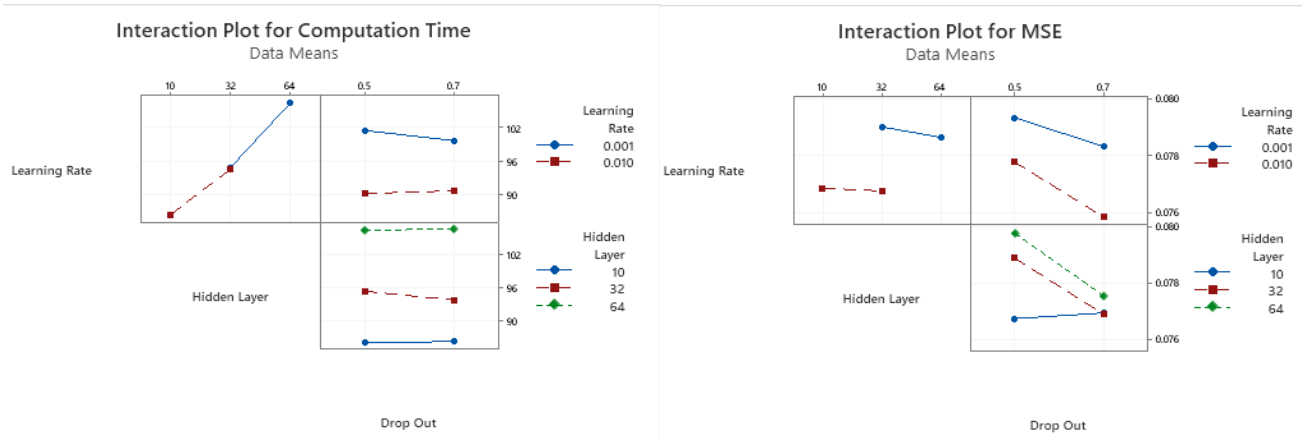

Figure 5: Interaction Plot of GNN Hyper Parameters

Using the risk calculation detailed in previous sections and tunned hyper parameters, the study creates a heat map

of risk levels for the default supply chain time series prediction. The risk prediction values on the heat map shows the risk of failure over a 13 day period (Figure 6). The heat maps show the risk values for each supply chain entity where green is little to no risk, yellow medium risk, and red is considerable risk.

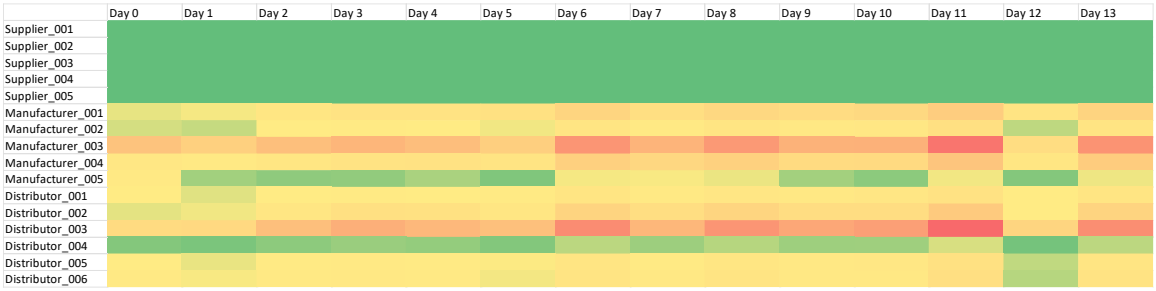

Figure 6: Risk Values for Nodes.

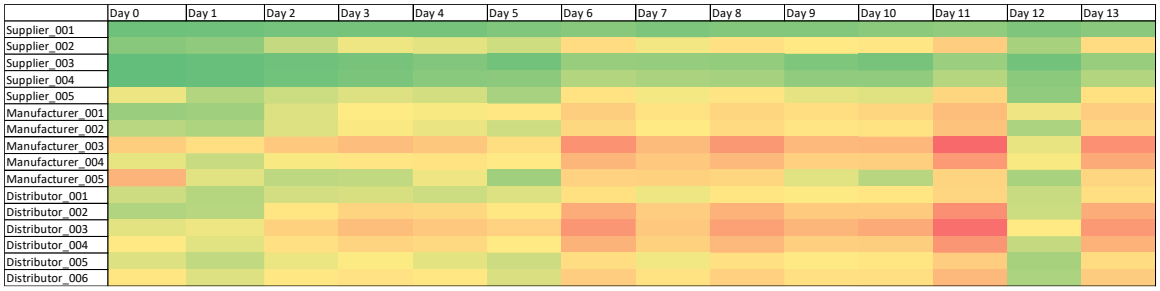

Figure 7: Katz Centrality Risk Calculation.

Figure 6 shows the risk values per node with between as a centrality measure, while Figure 7uses the Katz value instead for its centrality. The betweenness shows that the suppliers are not at risk. The reason for this conclusion is that the suppliers do not appear on the shortest path, leading to the conclusion that suppliers are interchangeable. The Katz values show a clearer picture of risk because suppliers have a risk value based on their importance to the nodes after them in the graph. The Katz centrality gives a better risk value then between. The reason is that Katz gives a localized importance of a node while between gives a nodes importance to a whole graph or subgraph. Understanding localized importance can be helpful in finding weak points.

CONCLUSIONS

This study viewed a supply chain as a network that products, goods, or services flow from node to node. Viewing a supply chain as a network flow allows researchers the ability to analyze supply chains using network analysis. Expanding on the concept that a supply chain is like a network, this study found that the underling structure of a supply chain is like a graph. A graph database can store the supply chain data as a graph structure, and storage as a graph structure allows for quicker analysis of data by bypassing the step to restructure the data. The graph data structure is input for a graph neural network, which can understand the structure, position, dependance and relationship metadata that is a part of the graph structure. The data needs to be encoded for use by the neural network and is done so using an MPNN which uses adjacency, node feature, and neighbor information to create a single vector encoding for each node within the graph. The GAT with centrality+LSTM model encoded and predicted future capacity values used to calculate risk of failure values. The results of GNN were compared to a validation set and a mean square error was used to rate accuracy. The GNN data, ARIMA, centrality, and Bayesian network influence were combined to create a risk of failure value for each node.

This study presents a who listic understanding of the data and can make better predictions of future graph data and

provides excellent results on the dataset; however, there is room for improvement. More specifically, the model needs a larger dataset to show its ability to scale. Future research could show how the model could handle a dynamic graph, since a supply chain changes shape. The graph structure would change over time which would change node influence. This study graph has a static influence map that does not change over time. Centrality of a node with a graph would also change since the graph would have a dynamic structure at specific time steps. A dynamic graph structure would better represent real world supply chains. Nodes and edges disappearing and reappearing would represent a supply chain entity going offline. The reason for a node going offline could be from outside influences or a node critical failure. The augmentation of the GAT is very simplistic in nature. A future study could use an equation to better incorporate centrality into the attention function. LSTM may not be the best neural network model for predicting future values. Future studies could investigate replacing the LSTM with other models to see if they produce better prediction values. Overall, this study concluded the concept of using a graph neural network to evaluate and understand a supply chain network is valid and useful to the industry.

REFERENCES

1. *Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., &Monfardini, G. (2008). The graph neural network model. IEEE transactions on neural networks, 20(1), 61-80.*
2. *Kosasih, E. E., &Brintrup, A. (2022). A machine learning approach for predicting hidden links in supply chain with graph neural networks. International Journal of Production Research, 60(17), 5380-5393.*
3. *Perozzi, B., Al-Rfou, R., &Skiena, S. (2014, August). Deepwalk: Online learning of social representations. In Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 701-710).*
4. *Xu, K., Hu, W., Leskovec, J., &Jegelka, S. (2018). How powerful are graph neural networks?.arXiv preprint arXiv:1810.00826.*
5. *Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., & Dahl, G. E. (2017, July). Neural message passing for quantum chemistry. In International conference on machine learning (pp. 1263-1272). PMLR.*
6. *Johnson, R., & Zhang, T. (2007). On the effectiveness of Laplacian normalization for graph semi-supervised learning. Journal of Machine Learning Research, 8(7).*
7. *Kipf, T. N., & Welling, M. (2016). Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907.*
8. *Hamilton, W., Ying, Z., &Leskovec, J. (2017). Inductive representation learning on large graphs. Advances in neural information processing systems, 30.*
9. *Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., & Mei, Q. (2015, May). Line: Large-scale information network embedding. In Proceedings of the 24th international conference on world wide web (pp. 1067-1077).*
10. *Defferrard, M., Bresson, X., &Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering. Advances in neural information processing systems, 29.*
11. *Grover, A., &Leskovec, J. (2016, August). node2vec: Scalable feature learning for networks. In Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 855-864).*
12. *Gasteiger, J., Bojchevski, A., &Günnemann, S. (2018). Predict then propagate: Graph neural networks meet personalized pagerank. arXiv preprint arXiv:1810.05997.*

13. Wu, X. M., Li, Z., So, A., Wright, J., & Chang, S. F. (2012). Learning with partially absorbing random walks. Advances in neural information processing systems, 25.

14. Li, Q., Wu, X. M., Liu, H., Zhang, X., & Guan, Z. (2019). Label efficient semi-supervised learning via graph filtering. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 9582-9591).

15. Chen, J., Ma, T., & Xiao, C. (2018). Fastgcn: fast learning with graph convolutional networks via importance sampling. arXiv preprint arXiv:1801.10247.

16. Seo, Y., Defferrard, M., Vandergheynst, P., & Bresson, X. (2018). Structured sequence modeling with graph convolutional recurrent networks. In Neural Information Processing: 25th International Conference, ICONIP 2018, Siem Reap, Cambodia, December 13-16, 2018, Proceedings, Part I 25 (pp. 362-373). Springer International Publishing.

17. Chen, J., Wang, X., & Xu, X. (2022). GC-LSTM: Graph convolution embedded LSTM for dynamic network link prediction. Applied Intelligence, 1-16.

18. Panagopoulos, G., Nikolentzos, G., &Vazirgiannis, M. (2021, May). Transfer graph neural networks for pandemic forecasting. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 35, No. 6, pp. 4838-4845).

19. Chen, D., Sain, S. L., & Guo, K. (2012). Data mining for the online retail industry: A case study of RFM model-based custmer segmentation using data mining. Journal of Database Marketing & Customer Strategy Management, 19, 197-208.

20. Tan, S. C., & Lau, J. P. S. (2013, December). Time series clustering: A superior alternative for market basket analysis. In Proceedings of the First International Conference on Advanced Data and Information Engineering (DaEng-2013) (pp. 241-248). Singapore: Springer Singapore.

21. Mahmud, S. (2020). Bangladesh COVID-19 daily cases time series analysis using facebook prophet model. Available at SSRN 3660368.

22. Kufel, T. (2020). ARIMA-based forecasting of the dynamics of confirmed Covid-19 cases for selected European countries. Equilibrium. Quarterly Journal of Economics and Economic Policy, 15(2), 181-204.

23. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., &Bengio, Y. (2017). Graph attention networks. arXiv preprint arXiv:1710.10903.

24. Brody, S., Alon, U., &Yahav, E. (2021). How attentive are graph attention networks?.arXiv preprint arXiv:2105.14491.

25. Zhu, X., Lin, Y., He, Y., Tsui, K. L., Chan, P. W., & Li, L. (2022). Short-Term Nationwide Airport Throughput Prediction With Graph Attention Recurrent Neural Network. Frontiers in Artificial Intelligence, 105.

26. Busbridge, D., Sherburn, D., Cavallo, P., &Hammerla, N. Y. (2019). Relational graph attention networks. arXiv preprint arXiv:1904.05811.

27. Li, C., Zhang, J., Wang, Y., & Liu, X. (2022). CAGAT: centrality-adjusted graph attention network for active scientific talent discovery. Personal and Ubiquitous Computing, 1-8.

28. Zhang, S., &Xie, L. (2020, July). Improving attention mechanism in graph neural networks via cardinality preservation. In IJCAI: Proceedings of the Conference (Vol. 2020, p. 1395). NIH Public Access.

29. Zhang, S., &Xie, L. (2020, July). Improving attention mechanism in graph neural networks via cardinality preservation. In IJCAI: Proceedings of the Conference (Vol. 2020, p. 1395). NIH Public Access.

30. Rozemberczki, B., Scherer, P., He, Y., Panagopoulos, G., Riedel, A., Astefanoaei, M., ... & Sarkar, R. (2021, October). Pytorch geometric temporal: Spatiotemporal signal processing with neural machine learning models. In Proceedings of the 30th ACM

International Conference on Information & Knowledge Management (pp. 4564-4573).

31. *Ebert-Uphoff, I. (2007). Measuring connection strengths and link strengths in discrete Bayesian networks. Georgia Institute of Technology.*
32. *Ebert-Uphoff, I. (2009). Tutorial on how to measure link strengths in discrete Bayesian networks. Georgia Institute of Technology.*
33. *Silva, N., Ferreira, L. M. D., Silva, C., Magalhães, V., & Neto, P. (2017). Improving supply chain visibility with artificial neural networks. Procedia Manufacturing, 11, 2083-2090.*
34. *Husna, A., Amin, S. H., & Shah, B. (2021). Demand forecasting in supply chain management using different deep learning methods. In Demand forecasting and order planning in supply chains and humanitarian logistics (pp. 140-170). IGI Global.*
35. *Kara, M. E., Fırat, S. Ü. O., & Ghadge, A. (2020). A data mining-based framework for supply chain risk management. Computers & Industrial Engineering, 139, 105570.*
36. *Teuteberg, F. (2008). Supply chain risk management: A neural network approach. Strategies and tactics in supply chain event management, 99-118.*
37. *Kosasih, E. E., Margaroli, F., Gelli, S., Aziz, A., Wildgoose, N., &Brintrup, A. (2022). Towards knowledge graph reasoning for supply chain risk management using graph neural networks. International Journal of Production Research, 1-17.*
38. *Hagberg, A., Swart, P., & S Chult, D. (2008). Exploring network structure, dynamics, and function using NetworkX (No. LA-UR-08-05495; LA-UR-08-5495). Los Alamos National Lab.(LANL), Los Alamos, NM (United States).*
39. *Ahmed, A. Kaleel, C. B. Senthilkumar, And S. Nallusamy. "Study On Environmental Impact Through Analysis Of Big Data For Sustainable And Green Supply Chain Management." International Journal Of Mechanical And Production Engineering Research And Development 8.1 (2018): 1245-1254.*
40. *Kiran, R., And Sd Sivakumar. "A Scale To Measure The Attitude Of Farmers Towards Vegetable Supply Chain Management." International Journal Of Educational Science And Research (IJESR) 7 (2017): 81-88.*
41. *Sathish, T. S. J. J., And J. Jayaprakash. "Optimizing Supply Chain In Reverse Logistics." International Journal Of Mechanical And Production Engineering Research And Development 7.6 (2017): 551-560.*
42. *Bhangu, Navneet Singh, And Sonia Grover. "Risk Assessment Of Gear Box Of Wind Turbine Using Fmea Approach." International Journal Of Mechanical And Production Engineering Research And Development 9.6 (2019): 725-734.*