Predicting Trajectories' Parameters Using Graph Convolutional Neural Networks

Ioannis Ioannidis

University of Piraeus
NCSR Demokritos
Athens

13-10-2023

DEPARTMENT OF DIGITAL SYSTEMS

NCSR DEMOKRITOS
INSTITUTE OF INFORMATICS AND TELECOMMUNICATIONS

Outline

Contents

- Introduction
- Background
- Methodology
- Dataset
- Experiments
- Conclusions

Introduction - Brief Description

- Recent advancement of aviation industry has increased the research interest in the field of Air Traffic Management (ATM).
- A contemporary problem in ATM field is the prediction of aircrafts' trajectories' parameters.
- Two of the most important such parameters are **Cost Index (CI)** and **Maximum Takeoff wight (MTOW)**, which prove to be really important for timescale and financial scheduling.

Introduction - Cost Index

- **Cost Index** is defined as the ratio of the time cost of an operating aircraft divided by the fuel cost.

$$\text{CI} = \frac{\text{Time Cost} \sim \$/\text{hr}}{\text{Fuel Cost} \sim \text{cents/lb}}$$

- CI practically defines the flight duration - gas savings tradeoff.
- The bigger the CI values, the bigger outgoings per hour, leading to speeding up decisions, which indicates more gas consumption.

Introduction - Maximum Takeoff Weight

- **Maximum Takeoff Weight** is defined as the maximum weight at which an aircraft meets all the airworthiness requirements in order to takeoff.
- Mathematically

$$\text{MTOW} = M\rho_a|V_0|^2$$

,where:
- M stands for aircraft's engineering properties
- ρ_a is air density
- $|V_0|$ is the takeoff speed

Introduction - Maximum Takeoff Weight

Another expression for **Maximum Takeoff Weight** is

$$\text{MTOW} = \text{MTOW}_0 \times C_f$$

,where:

- MTOW_0 is the value set for MTOW at fixed conditions
- C_f is a corrector for the existing circumstances at each time

Introduction - Problem Statement

- The combination of CI and MTOW parameters is capable of predicting flight's Key Performance Indicators (KPIs), such as fuel needed, time spended and distance covered during a flight.
- Hence, airline companies prefer to keep CI and MTOW parameters secret in concepts of their commercial policy.
- In that way, many research works in ATM field prefer to manually set them, instead of consuming demands for estimating them.
- This perspective affects the final outcomes and leads to uncertain results.

Introduction - Proposed Solution

- This work aims on creating models for the accurate prediction of CI and MTOW.
- This is achieved by using simulated trajectories data and casting the problem as a regression task.
- The method proposed is based on the integration of Convolutional Neural Networks (CNNs) and spatial graph theory.
- The Graph Convolutional Neural Network (GCNN) focalizes on the scemantic process of the data, the graph structure and hyperparameters tuning.

Background - MLP

- **Multilayer Perceptron (MLP)** is the most basic form of deep neural networks, constituting by sets of layers, each one containing a number of identical neuron units.
- Units are connected with each other via bias weights in a feedforward way.
- MLPs work as a function aproximator in a set of input and output vectors.

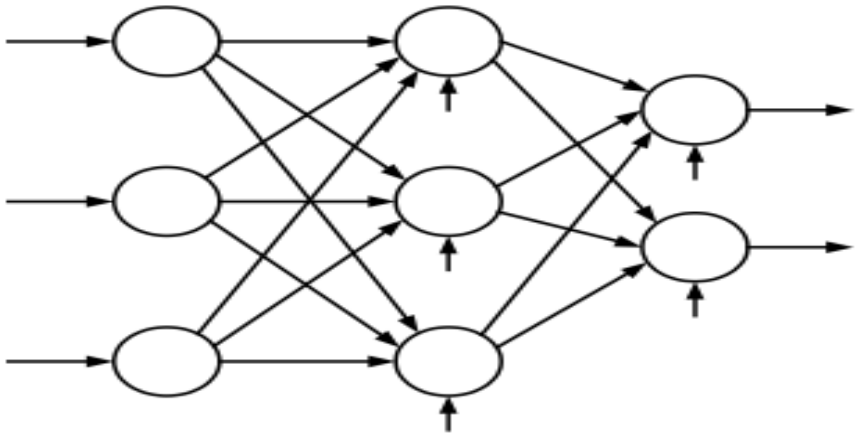


Figure 1: Fully Connected Multilayer Perceptron

Background - CNN

- The neurons of a CNN are organised into three dimensions, receiving $h \times w \times d$ dimensional inputs.
- A typical CNN is made out of three (3) types of parts:
  - convolutional layers combined with an activation function
  - pooling layers
  - fully-connected layers.

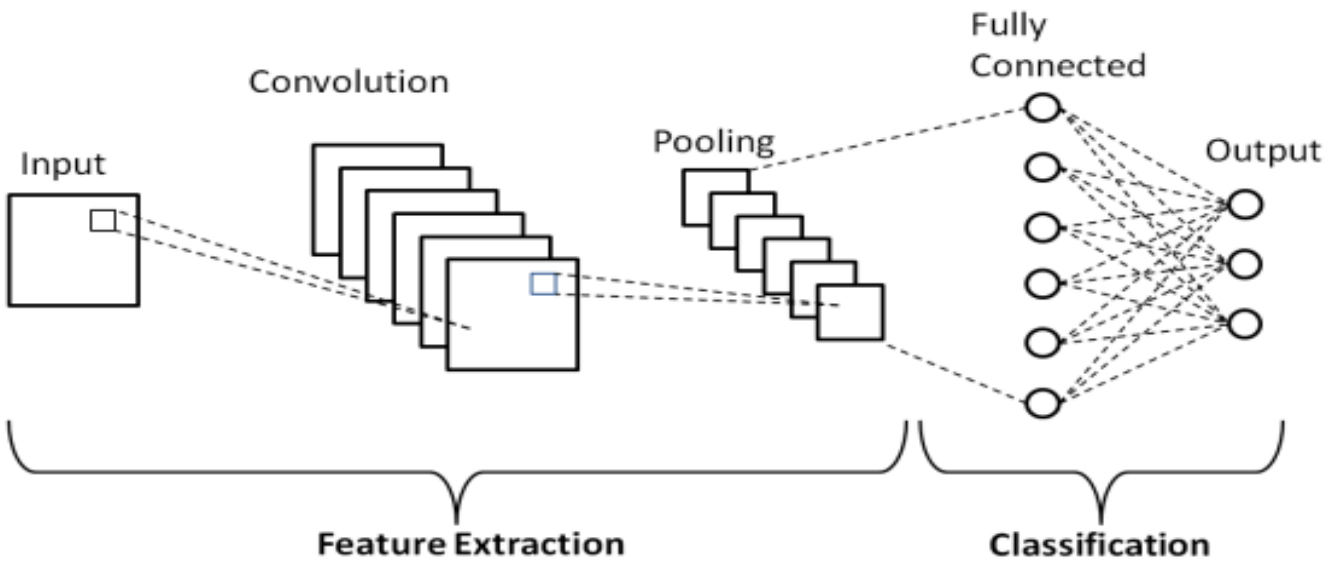


Figure 2: Simple Convolutional Neural Network

Background - CNN

- The convolution is basically a feature extraction operation, where a small matrix called kernel, is applied across the input matrix in a tensor shape, computing a Hadamard product.
- Convolution's outcome is, then, passed through an activation function, which is about the feature of that outcome to be retained and mapped out by a non-linear function

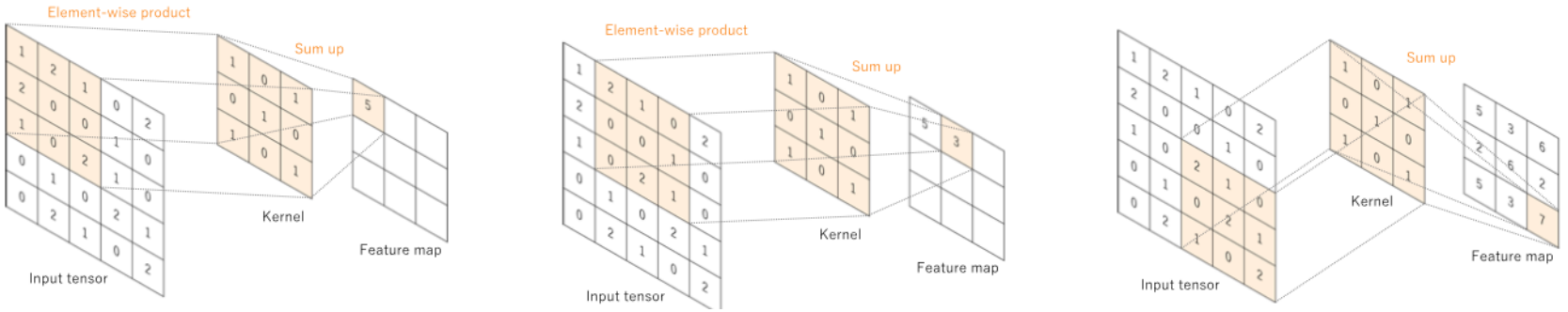


Figure 3: Convolution Operation

Background - CNN

- The main function of a pooling layer is the downsampling of the feature maps, created by the convolution process.
- It shrinks the large-sized convolution's outcome to smaller ones, by maintaing the majority of the important information at every pooling step.

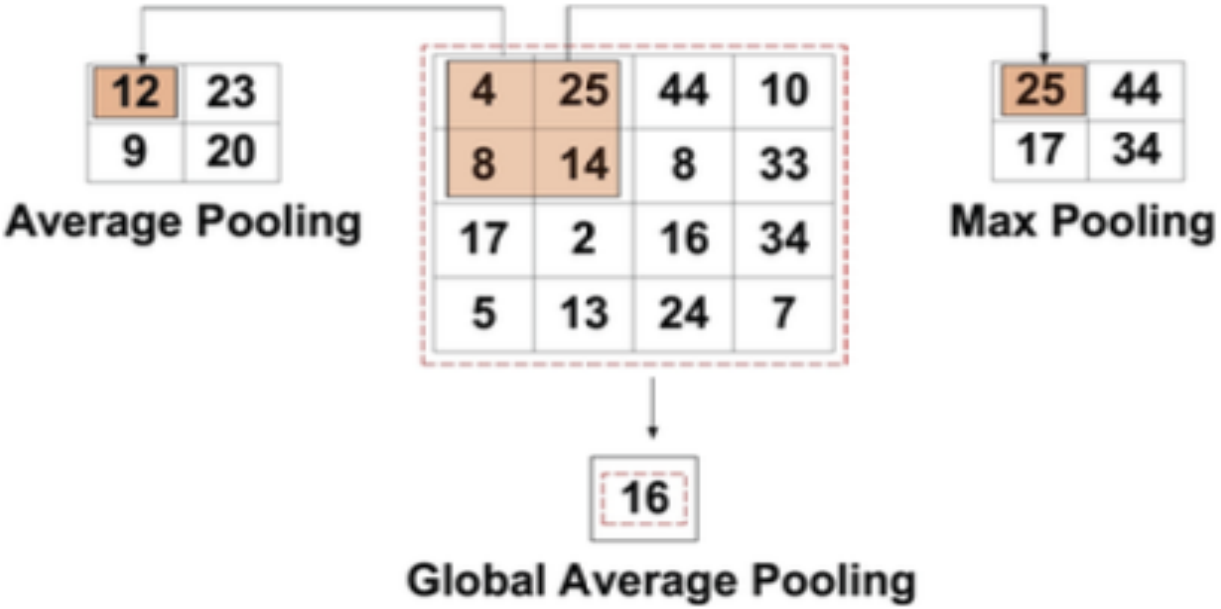


Figure 4: Common Types Of Pooling Operations

Background - CNN

- The outcome feature maps of the last convolution or pooling layer are flattened and converted to vectors which are being passed through the fully connected layer.
- All neurons are connected with each other by learnable weights and contribute to produce the final outcome.
- When the network deals with a classification task, this outcome is probabilities for each class, while on the other hand in regression tasks it is a real numerical value.

Background - Graph Convolution

- Graph convolution is the extention of CNNs in the graph domain. GCNNs are splitted into two basic domains, depending either the graph form is spatial or spectral.
- Spectal convolution performs an eigen decomposition of the Laplacian matrix of the graph, helping in understanding the underlying structure of the graph and identify clusters/sub-groups.
- Spatial convolution works on local neighbourhood of nodes and understands the properties of a node based on its k local neighbours.

Background - Transformers

- Transformers are deep learning models able to differentially calculate and take into account the significance of each part of the input data.
- This is achieved with the adoption of mathematical formulas estimating, at each timestep, the importance level of each part of an input sequence, known as attention mechanisms.
- Attention mechanisms make use of three basic components:

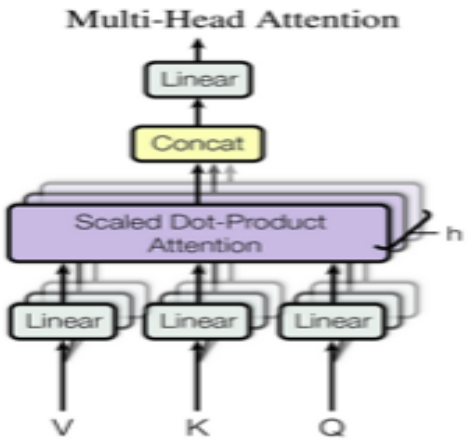


Figure 5: Multi-Head Attention Mechanism

Background - Transformers

- Q (Query) matrix contains the vector representation of the focused input, which will be used to compare against other inputs in the sequence.
- K (Key) matrix contains the vector representation for other inputs in the sequence, which are used for the similarity measurement of the target input (using query vector) and other inputs.
- V (Value) matrix contains the value vectors for all the inputs in the sequence, holding the contextual information for each of them.
- Then the weighted sum of the value vectors is calculated, using weights determined by the similarity scores, securing that the final contextual representation is affected more by the relevant inputs.

Background - Transformers

- The transformers architecture is based on the encoder - decoder form.
- Encoders generate transformed information concerning the parts of the inputs that are relevant to each other.
- Decoder layers receive all the encodings and using their incorporated contextual information, they generate an output sequence, with the usage of the attention mechanisms.

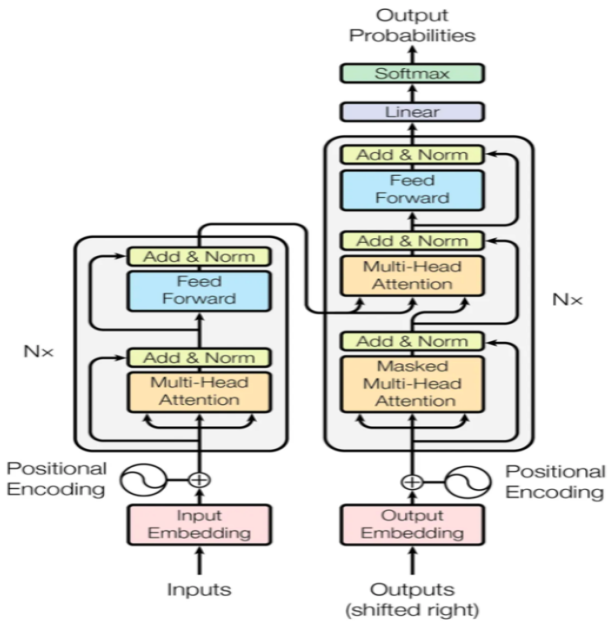


Figure 6: The Transformer-model architecture

Methodology - Initiation

- The *CI* and *MTOW* formulation problem is simulated in an agents collaboration environment.
- The environment is represented as a graph and each agent as a node.
- The communication of two agents is imprinted on the graph with the connection of their respective nodes via edges.
- Graph structure is passed through a convolution, using multiheads attention as the convolutional kernel.
- In that way, graph convolution extracts the relation representation between agents and convolutes the features from the neighboring ones.

Methodology - Preliminaries

Given a graph $G = (V, E)$, where V denotes the set of nodes and E the set of edges, a GCNN receives as input:

- a feature matrix (F), $N \times L^0$ sized, where N is the number of nodes (i.e agents) and L is the number of feature inputs for each node
- a $N \times N$ matrix representation of the graph structure, called Adjacency matrix ($A(i, j)$). Here, i and j stand for ith row and jth column respectively and $A(i, j) = 1$, when there is an edge between node i and node j and 0, otherwise

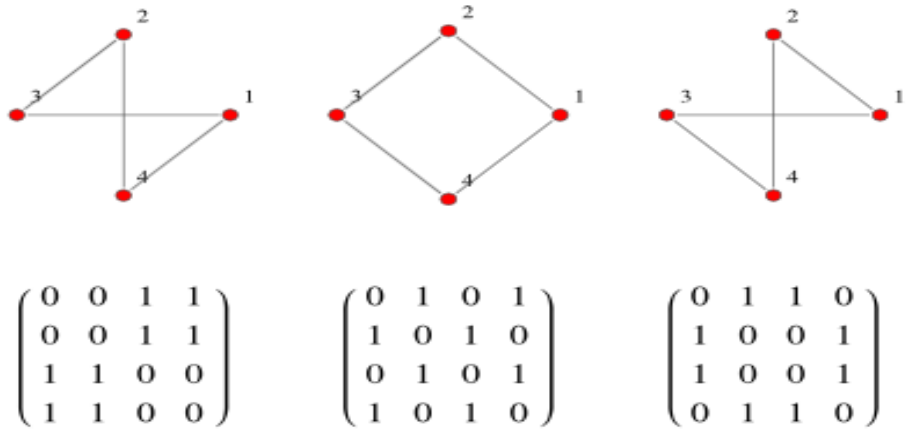


Figure 7: Adjacency matrices examples

Methodology - Preliminaries

- Hence, a hidden layer of a GCNN can be expressed as

$$H^i = f(H^{i-1}, A)$$

,where:
- $H^0 = F$
- f is a propagation rule.

- Each layer H^i consist a $N \times F^i$ feature representation for each node.
- Those features are combined and produce as output the next layer's features using the propagation rule f, until this procedure reaches the last layer, which provides the final outcome.

Methodology - Architecture

The GCNN proposed consists of three modules:

- MLP Encoder
- Two (2) Convolutional Layers
- Fully Connected Layer

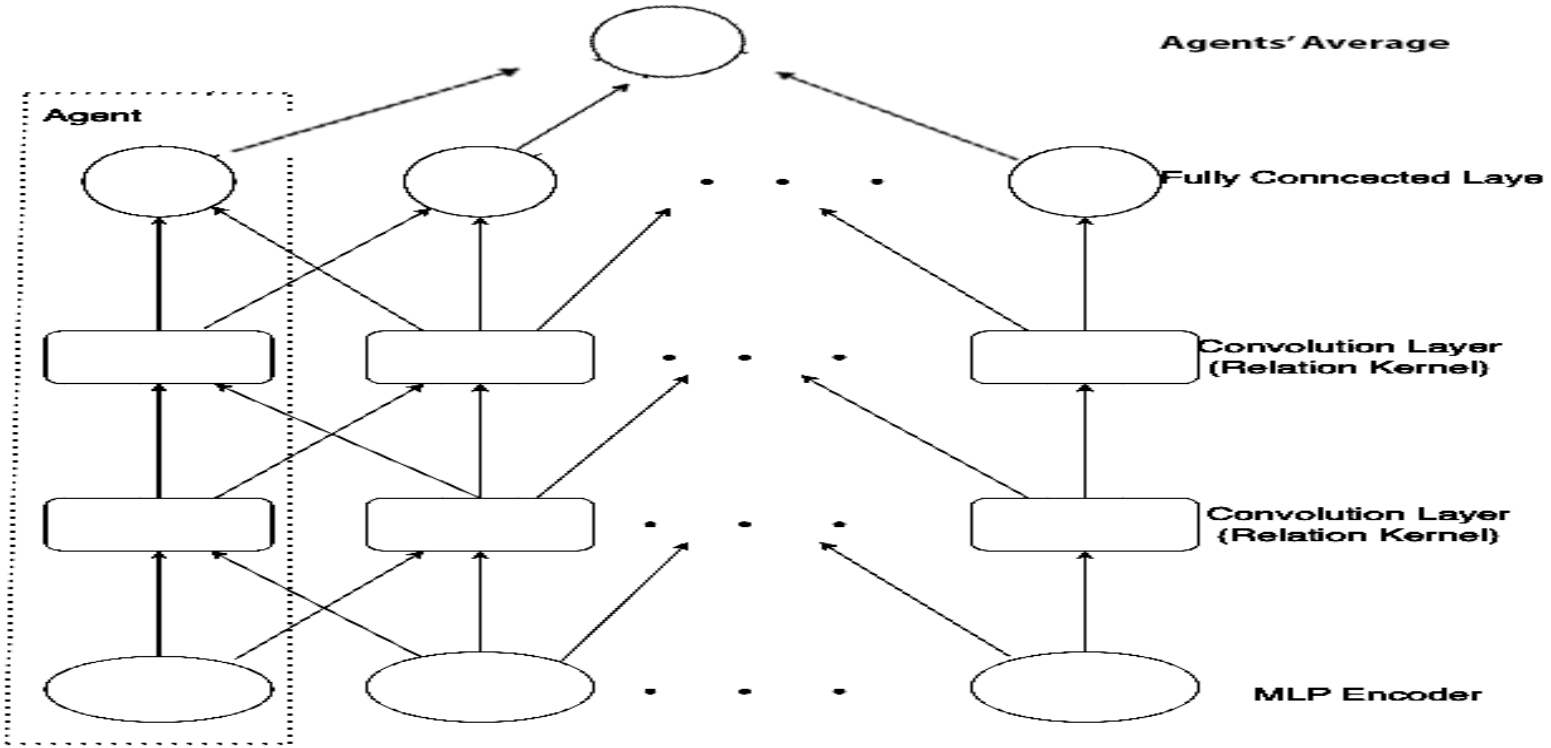


Figure 8: GCNN Architecture

Methodology - Architecture

- Each agent receives an input o_i^y, corresponding to its local observation, part of the whole observation y.
- This local observation is passed through a two layered MLP and encoded into a feature vector h_i^y.
- Convolutional layer receives the feature vectors of local region agents (i.e of agent i and its neighbors B_i) and integrates them to produce the updated feature vector $h_i'^y$.

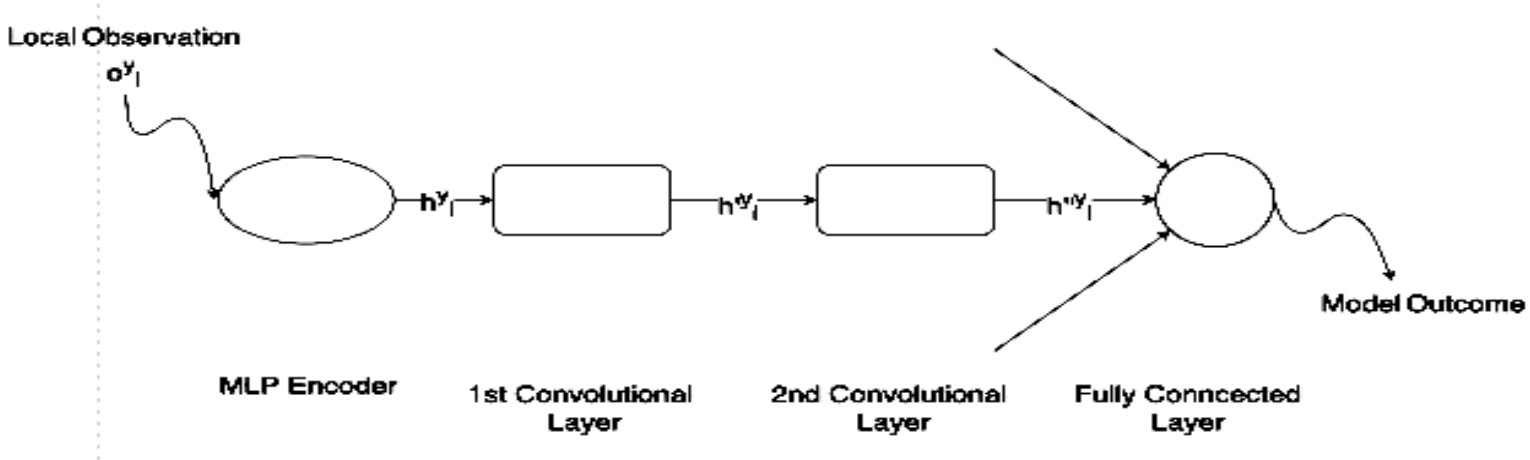


Figure 9: Observation Pass Through GCN

Methodology - Architecture

- That one-hop convolution procedure allows agent i to communicate with its first-order neighbours.
- By stacking n convolution layers, agent i can aggregate information from nodes that are up to n-hops away.
- Finally, at the fully connected layer all $h_i''^y$ for each agent i are concatenated into one vector and passed through three (3) dense layers produce the model's outcome.

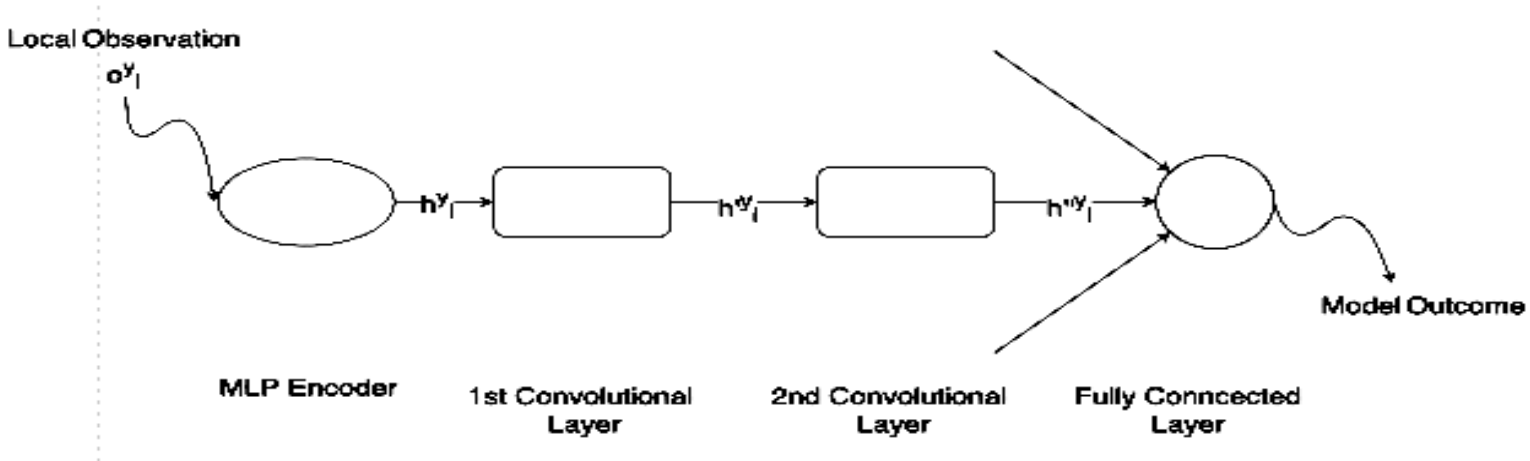


Figure 10: Observation Pass Through GCN

Methodology - Relation Kernel

- Interactions between agents are computed using multi-head dot product attention (MHDPA) as the convolution kernel.
- MHDPA takes into account scaled dot-product attention multiple times in parallel, combining the outputs of independent attentions by concatenating and transforming them into the expected dimensions.
- This gives the model the attribute to attend observations from different representation subspaces at different positions.
- Calculations are, then, combined together to produce the desired multi-head attention score, which makes the transformer able to encode multiple relationships with great performance.

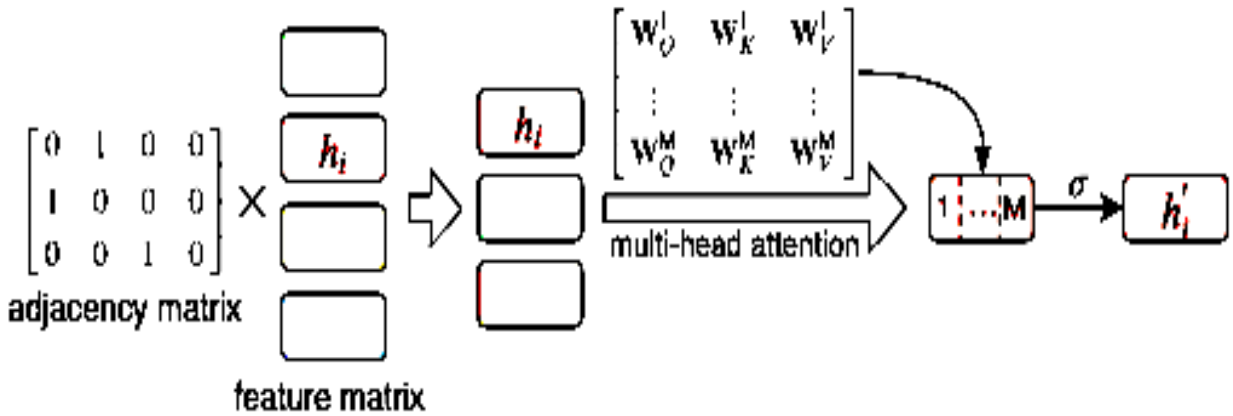
Methodology - Relation Kernel

- MHDPA projects the input feature of each agent i to Q, KV and V matrices by each independent attention head m.

$$a_{ij}^{m} = \frac{exp(\tau W_Q^m h_i^m \cdot (W_K^m h_j^m)^T)}{\sum_{j \in B_{+i}} exp(\tau \cdot W_Q^m h_i^m \cdot (W_K^m h_k^m)^T)}$$

- The output of the convolutional layer is produced with the concatenation of the weighted input features through σ

$$h_i' = \sigma(\text{concatenate}[\sum_{j \in B_{+i}} a_{ij}^m W_V^m h^j, \forall m \in M])$$



Dataset - Description

- The dataset used comprises simulated trajectories of flights, provided by DYNAMO_FP, combining the provided CI and MTOW values in all possible combinations.
- Cost index is ranging from zero (0) to one hunded (100), with a two (2) step interval, while take-off weight is ranging from zero point six (0.6) to one (1), with a zero point one (0.1) step interval.
- Thus, for each flight two hundred and fifty (250) (50 CI $\times$ 5 MTOW values) combinations are provided.
- Each trajectory file comprised about six hundred (600) timestamped points, each point enhanced with the variables provided by DYNAMO.

Dataset - Description

- For the experimental procedure we will be using two distinct datasets: DYNAMO_FP(69) and DYNAMO_FP(51), distinguished by the variables provided for each timestamp point.
- DYNAMO_FP(69) includes variables provided by $DYNAMO_{optimization}$ mode
- DYNAMO_FP(51) results of preprocessing methods applied in DYNAMO_FP(69), perusing only its spatiotemporal details, considering it included information which was not easily accessible.

Dataset - DYNAMO_FP(69) Variables

- Variables provided by DYNAMO_FP(69) are:

Variable	**Description**
h[ft]	geometric altitude
Temp[ºC]	air temperature
Press[hPa]	air pressure
Wn[kt]	North wind component
We[kt]	East wind component
Ws[kt]	Along path wind component
Wx[kt]	Cross-wind component
Lat[o]	Latitude
Lon[o]	Longitude
vdot	Derivative of True Airspeed
hdot	Derivative of geometric altitude

Table 1: DYNAMO_FP(69) Variables Description

Dataset - DYNAMO_FP(51) Variables

- Variables provided by DYNAMO_FP(51) are:

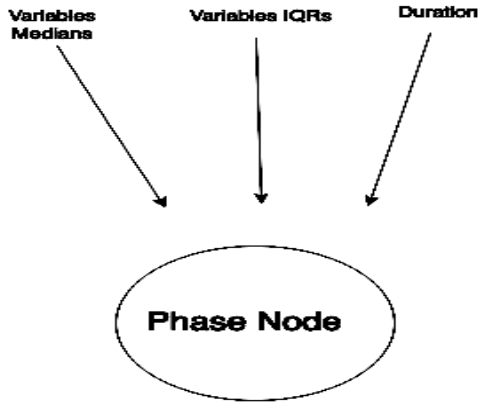
Variables	**Description**
h[ft]	geometric altitude
Temp[oC]	air temperature
v-Wn[kt]	v wind component
u-We[kt]	u wind component
Lat[o]	Latitude
Lon[o]	Longitude
vdot	Derivative of speed
hdot	Derivative of geometric altitude

Table 2: DYNAMO_FP(51) Variables Description

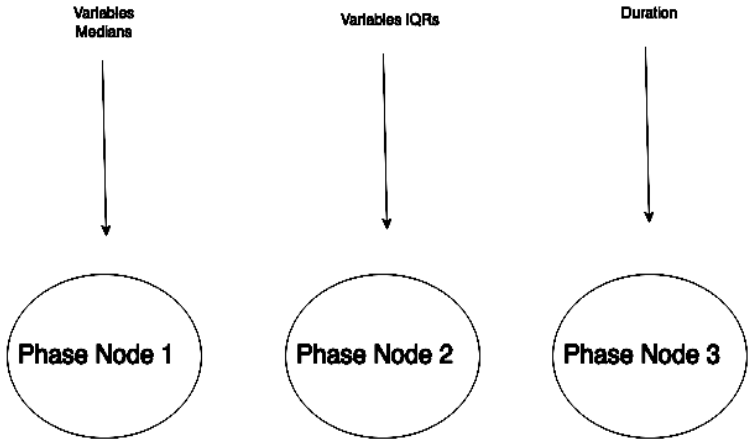
Experiments - Description

- This work investigated two aspects of the methodology, in terms of configurating the overall algorithm's architecture and optimizing the performance:
  - the number of agents contributing
  - the input features for each of them
- For the examination of the agents' number contributing we only need to experimentise between the number of nodes and the way they construct the graph.
- For the input features examination we explore the part of the trajectory that each of them processes

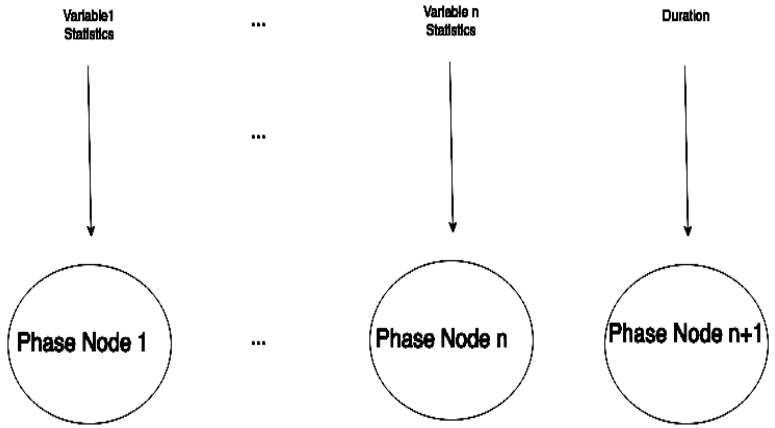
Experiments - Agent Wise Exploration



(a) All Variables Statistics

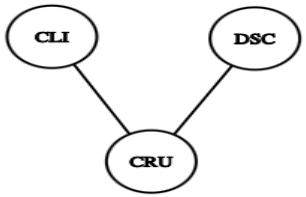


(b) Separated Statistics

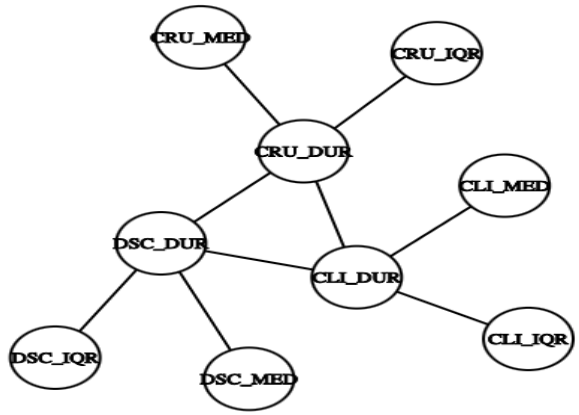


(c) Separated Variables

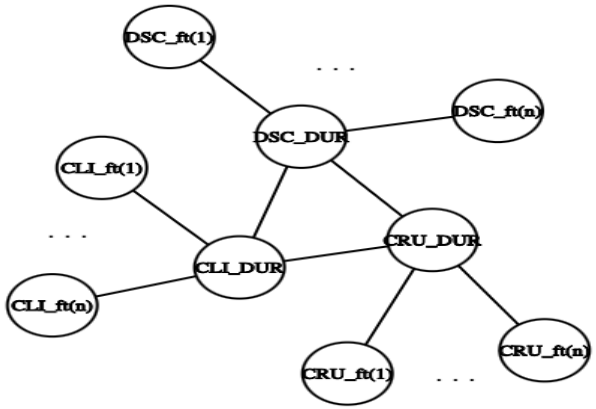
Experiments - Agent Wise Exploration



(a) All Variables Statistics



(b) Separated Statistics



(c) Separated Variables

Experiments - Feature Wise Exploration

- For this experimentation, we splitted each flight phase into subphases, studied each subphase separately and combined them back again to rebuild the original phase.
- The features for each of the timespans are preprocessed apart, statistical metrics and duration were calculated and then merged again to reconstruct the phase in a more detailed way.
- Considering a phase's altidute values fluctuate in the interval $[a, b]$, we classified each point between n equal sub-intervals (subphases). Each sub-interval has a length of $\delta = \frac{b-a}{n}$. Hence, the sub-intervals must be

$$[a, a+\delta], [a+\delta, a+2\delta], ..., [a+(n-1)\delta, b]$$

Experiments - Results

- DYNAMO datasets were previously used for training and testing several machine learning algorithms.

	DYNAMO_FP(51)		DYNAMO_FP(69)	
Method	CI MAE	MTOW MAE	CI MAE	MTOW MAE
GBM	3.65	0.022	2.91	0.009
ANN	4.38	0.023	3.89	0..016
KRR	4.05	0.021	4.06	0.026
SVR	3.98	0.03	3.94	0.021
LASSO	8.06	0.078	6.03	0.039

Table 3: Results on previous Methodologies

Experiments - Results

- Results for the proposed GCNN are presented in the abbreviation form AGxSBPHy, where x stands for the number of agents and y for the subphases that each flight phase is splitted.

	DYNAMO_FP(51)		DYNAMO_FP(69)	
Method	CI MAE	MTOW MAE	CI MAE	MTOW MAE
AG3SBPH0	5.83	0.031	4.92	0.027
AG9SBPH0	9.51	0.075	8.67	0.67
AGnSBPH0	17.94	0.128	15.78	0.107
AG3SBPH2	5.73	0.037	4.46	0.024
AG3SBPH3	4.89	0.029	3.96	0.020

Table 4: GCNN Results

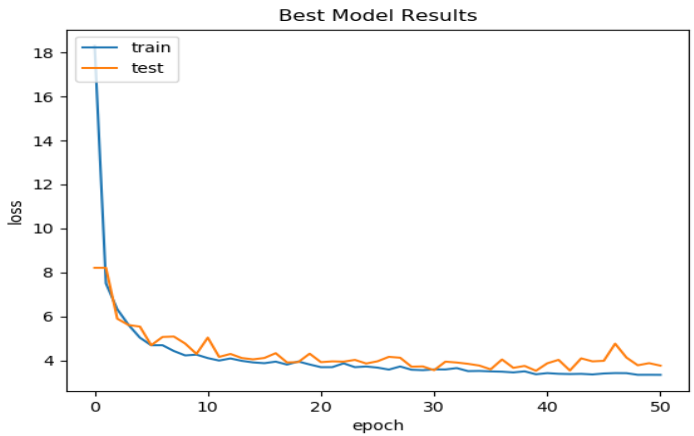
Experiments - Best Model's Performance

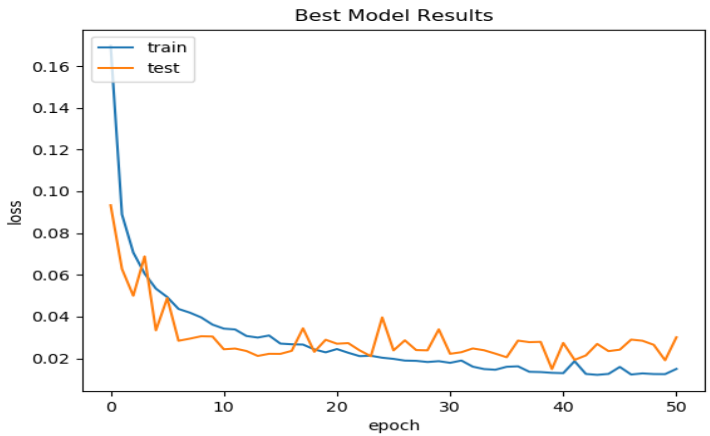
- With the addition of the decimal restriction, in terms of configurating training to to the numerical behavior of the target variables (0 decimals for CI and 1 decimal for MTOW), best model's final performance is imprinted as following

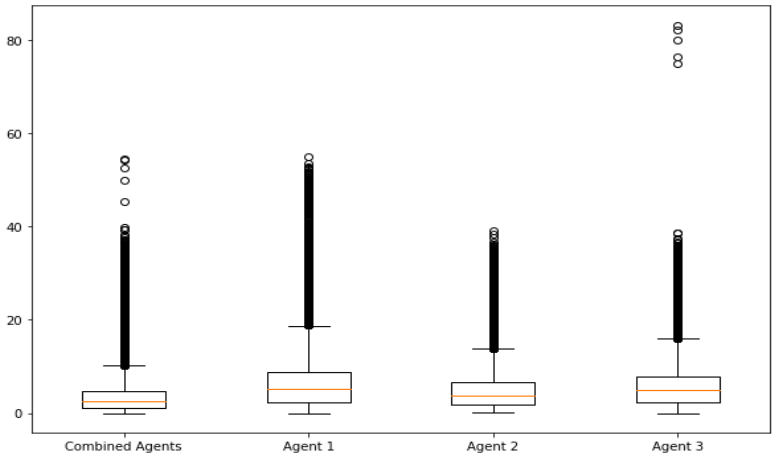
Dataset	CI MAE				MTOW			
	mean	std	IQR	range (max - min)	mean	std	IQR	range (max - min)
DYNAMO_FP(51)	4.59	5.36	5	44	0.020	0.052	0	0.4
DYNAMO_FP(69)	3.78	4.31	4	39	0.017	0.048	0	0.4

Table 5: Best Models MAE

Experiments - Results







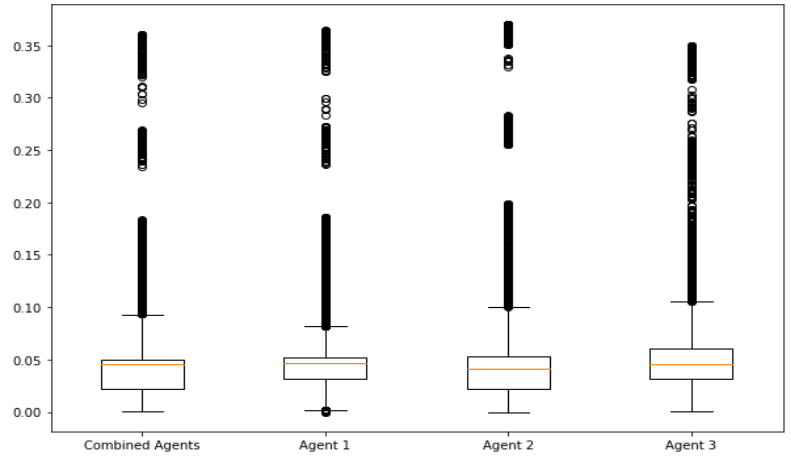
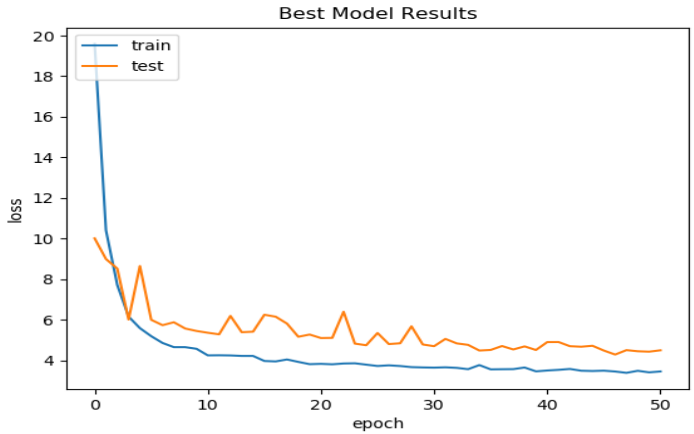
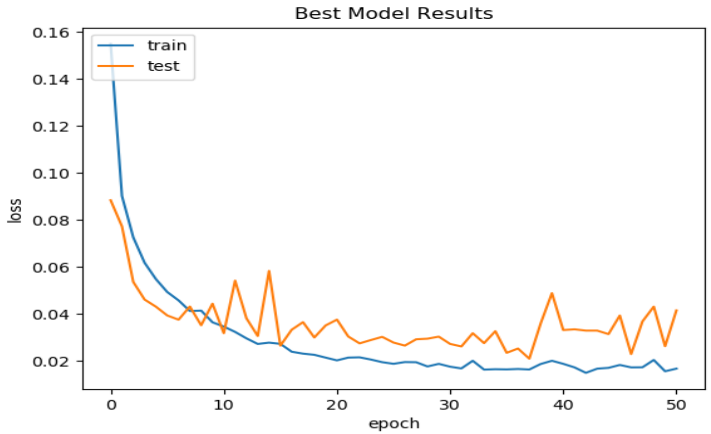
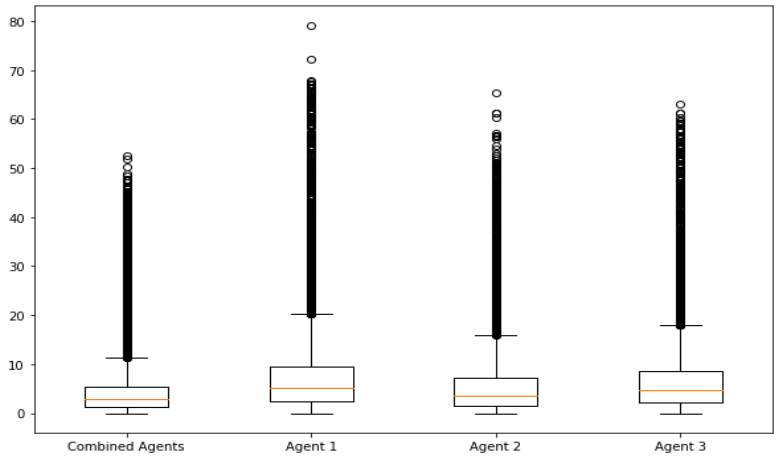


Figure 14: Learning Curves - MAE Boxplots for DYNAMO_FP(69)

Experiments - Results







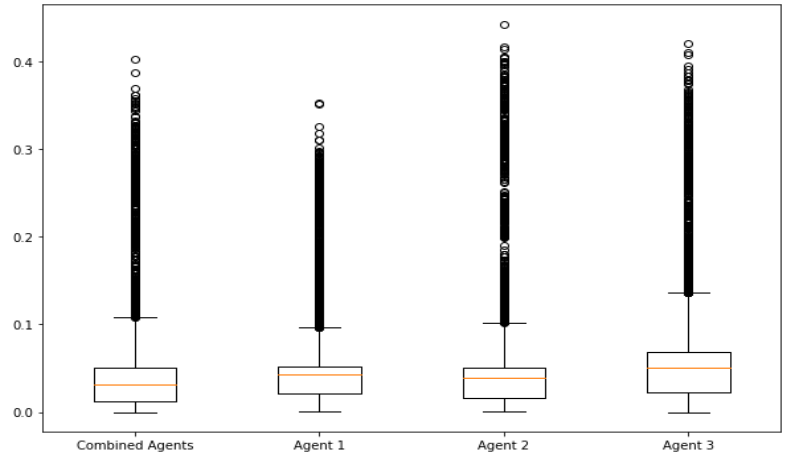


Figure 15: Learning Curves - MAE Boxplots for DYNAMO_FP(51)

Conclusions

- GCNN does not outperform the, leading, GBM algorithm's performance. However, it worthily competes all five approaches, being on top of LASSO and KRR.
- The structure proposed can be further be applied to many regression problems, with the suitable configuration of the input data.
- Models provided by GCNN can be further improved, with the suitable exploration of hyperparameters, such as different optimizers, dropout percentages and layers, weights initialization, number of hidden units, activation functions and loss functions.