SIGNN – Star Identification using Graph Neural Networks

Floyd Hepburn-Dickins[1], Mark W. Jones[1], Mike Edwards[1], Jay Paul Morgan[1], Steve Bell[2]
[1]Swansea University, United Kingdom
[2]UK Hydrographic Office, Taunton, Somerset, United Kingdom
{880940,m.w.jones,Michael.Edwards,J.P.Morgan}@swansea.ac.uk, Steve.Bell@UKHO.gov.uk

Abstract

As a solution for the lost-in-space star identification problem we present Star Identification using Graph Neural Network (SIGNN), a novel approach using Graph Attention Networks. By representing the celestial sphere as a graph data structure, created from the ESA's Hipparcos catalogue, we are able to accurately capture the rich information and relationships within local star fields. Graph learning techniques allow our model to aggregate information and learn the relative importance of the nodes and structure within each stars local neighbourhood to it's identification. This approach, combined with our parametric data-generation and noise simulation, allows us to train a highly robust model capable of accurate star identification even under intensive noise, outperforming existing methods. Code and generation techniques will be available on https://github.com/FloydHepburn/SIGNN.

1. Introduction

Celestial navigation is an offline alternative to a reliance on global positioning systems, which are vulnerable to potential threats such as satellite malfunctions or signal interference from third parties.

There are two primary problems, star detection, that is to pinpoint prospective stars within an image, and star recognition, the correct identification of a given star. The former is a widely solved issue [21, 30], whereas the latter remains a point of active research [18, 20].

Whilst the focus of modern research is in spacecraft and attitude determination [18, 23], the techniques and methods used can be applied to terrestrial navigation. Navigation requires the accurate pinpointing and recognition of stars. Modern pattern recognition techniques can vastly improve both star acquisition and identification and have been shown to do so when applied to attitude determination. We further examine their efficacy on systems grounded to the Earth's surface alongside the challenges that entails.

Machine learning's broad use in astronomy is well established, with the vast data involved being prime for machine learning data analysis [19]. Galaxy classification [22], star cluster classification [16], spectral analysis [1] and pulsar identification [4] are just a few examples. Recently, deep learning approaches to star recognition have made use of convolution neural network (CNNs) [14] and generative adversarial networks (GANs) [6], whilst examining performance within noisy environments.

Recent advances in Graph Neural Networks (GNN) demonstrate their ability to learn complex relationships in graph structured data [32]. In this paper we present our novel approach to star identification by applying GNNs to a graph representation of the celestial sphere. We combine this with our parametric data-generation and noise simulation to ensure a robust tolerance to noise.

2. Existing Work

Traditional star identification solutions are broadly divided into two sub-categories (sub-graph isomorphism and star pattern recognition). These classical approaches are well reviewed by the literature [8, 18, 20]. We also review the latest approaches to star recognition using machine learning techniques.

2.1. Sub-graph isomorphism

Sub-graph isomorphism techniques aim to match observed star configurations against pre-computed known patterns. The triangle algorithm [10] and it's variations [13] represent each star as a star triplet formed from it's nearest neighbours, with the intent to later match detected triplet structures against a pre-computed database. The pyramid technique [11] builds on this by including an additional star for more varied and detailed initial patterns. Hash mapping [27] is shown to enable faster retrieval and matching with the known configurations.

2.2. Star pattern recognition

This approach involves creating a star feature that represents each prospective star based on its spatial and intrinsic

properties. The grid algorithm [15] forms a pattern vector from the position between a target star and its neighbours on a pre-defined grid and is further improved [12] to address shortfalls with noise tolerance. More recently, [29,31] represented stars using radial patterns, described via polar coordinate systems and discretized into a pattern vector for matching.

2.3. Machine Learning within Star Recognition

Machine learning (ML) approaches have enabled a transition away from hand crafted star patterns, instead leveraging ML methods to autonomously generate and recognise star patterns using the inherent information in the data. RPN Net [28] applies a two-fold star pattern generator and classifier approach, utilising an auto-encoder/decoder [3] to generate a learned representation of each star tolerant to noise. Initial inputs are created from the discretized distances between a target star and its neighbours, concatenated with the discretized inter-neighbour distances, encoding the overall relationship for the neighbourhood and target star. Stars can similarly be represented via radial patterns for use in CNNs [17]. However these discretization processes can lead to a loss of finer detail between nodes, impacting performance.

Jiang *et al*. [9] transform star patterns into "spider-images". Connections are drawn between stars in a defined neighbourhood radius and colour encoded using their discretized distances, each bin being assigned a different colour. This image representation is fed into a two-stage hierarchical CNN for classification. Whilst this method converts spatial relationship into a visual format suitable for CNNs, it still suffers a loss of detail in the discretization and the resulting spider images may not fully capture the intricate geometrical relationships between stars. By representing star-fields as polar coordinates and applying a log-polar transform before discretization, it is possible to encode both distances and relative angles [26].

All these techniques result in a degradation of the initial representation before it is applied to the model. In using GNNs with graph structured data we are able to utilise the full information available in each stars neighbourhood.

3. Methodology

Our proposed method for star identification represents the celestial sphere as a graph data structure, allowing our GNN to model and learn the complex relationships between nodes (stars), capturing spatial distances, relational connections, and topological structure alongside the initial node features. GNNs are particularly well-suited to this task due to their ability to effectively process and learn from the intricate relationships within graph-based data, enabling accurate recognition of star configurations under varying degrees of noise and interference.

3.1. Graph Construction

We model the celestial sphere as an undirected complete graph $G = (V, E)$ where each node $v \in V$ represents a star and each edge $(u, v) \in E$ represents an angular distance between u and v. The graph is modified to include only those edges below a certain angular distance threshold (pattern radius), simulating FOV limitations in star sensors. This causes each star to have a neighbourhood $\mathcal{N}(i)$ derived from simulating sensor field of view (FOV) and star magnitude sensitivity (section 3.4). Our GNN is trained on noisy variations of this celestial graph with each prediction being based on a node's (star's) local neighbourhood information.

3.2. Source Detection

When considering an image of the night sky, potential stars are detected via a process of source detection [21, 30]. This produces the absolute positions of source centroids (measured in pixel coordinates) and their brightness levels (measured in pixel values). The angular separation between sources is calculated by mapping coordinates to a reference frame using arcseconds per pixel based on camera parameters. Brightness data is processed to give each source a magnitude relative to its local neighbourhood. This processed data is used to transform an image into its graph representation as in section 3.1. Each source becomes a node whose feature is its relative magnitude within the defined pattern radius. Each source connects via an edge to each other source within its pattern radius with the edge attribute storing the angular separation between each pair. The resulting image graph would be a subgraph of the celestial sphere graph and is used as input to our trained GNN to identify the nodes (stars) within.

3.3. Graph Attention Network

We employ a 2 layer graph attention network (GAT) [5, 24] to process this graph data. The GAT layers are implemented using PyTorch geometric [7]. By using a 2 layer GAT, the neighbourhood comprises a star's closest neighbours as outlined above and also their neighbours' neighbours similarly defined (section 3.4). The relative brightness of each star within its neighbourhood is assigned to its initial feature vector x_v^0. An initial linear feature transform expands dimensionality whilst at each GAT layer the feature vector is updated through the GAT process.

The transformation of a node's feature in our GAT layers using multiple attention heads is given by:

$$\mathbf{x}_i' = \Big\|_{h=1}^{h} \sigma \left(\sum_{j \in \mathcal{N}(i) \cup \{i\}} \alpha_{i,j}^h \mathbf{W}^h \mathbf{x}_j \right) \tag{1}$$

The attention coefficients $\alpha_{i,j}^h$ for each head h are computed as:

$$\alpha_{i,j}^{h} = \frac{\exp\left((\mathbf{a}^{h})^{\top}\,\mathrm{LeakyReLU}\left(\mathbf{W}_{s}^{h}\mathbf{x}_{i} + \mathbf{W}_{t}^{h}\mathbf{x}_{j} + \mathbf{W}_{e}^{h}\mathbf{e}_{i,j}\right)\right)}{\sum_{k\in\mathcal{N}(i)\cup\{i\}}\exp\left((\mathbf{a}^{h})^{\top}\,\mathrm{LeakyReLU}\left(\mathbf{W}_{s}^{h}\mathbf{x}_{i} + \mathbf{W}_{t}^{h}\mathbf{x}_{k} + \mathbf{W}_{e}^{h}\mathbf{e}_{i,k}\right)\right)} \tag{2}$$

Figure 1 shows the neighbourhood representation of Sirius after being passed through two GAT layers and its FOV. A final linear classification layer utilises this 2-hop representation to generate logits for each node within the input graph, corresponding to the probability distribution over the catalogued star IDs. These values are then used to determine the final prediction and confidence scores. Our GNN architecture is outlined in Figure 2.

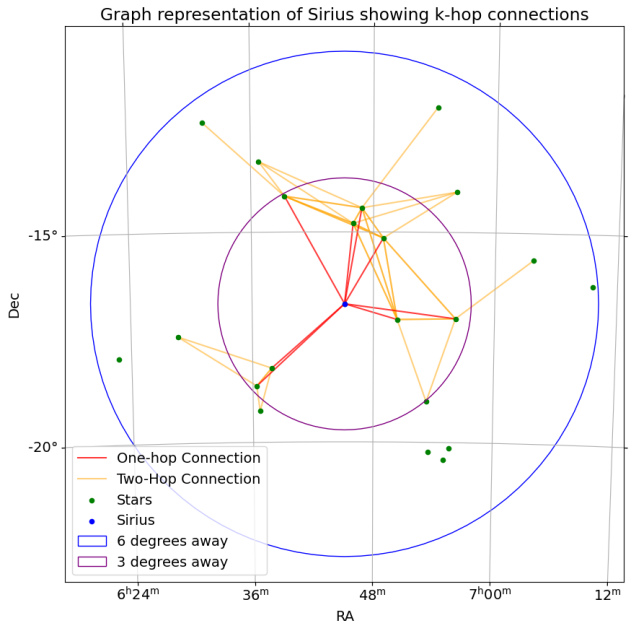

Figure 1. Example of the receptive field and messaging passing for the star Sirius within a two layer GAT network. Sirius is connected to its neighbours within a radius of 3° (red edges). Those neighbours are connected to their neighbours within a radius of 3° (orange edges).

3.4. Data Generation

We use the Hippacros star catalogue, which contains accurate RA/DEC coordinates and magnitude for each star, to create our reference celestial sphere graph. Coordinates are used to create an adjacency matrix of the angular separation between each star pair using the Vincenty formula [25]. Each node stores a star's data (RA/DEC, Vmag, ID) and is connected to it's neighbourhood $\mathcal{N}(i)$ with an edge whose attribute stores the angular separation between the connected star pair.

Similar to kernel density estimation, we could choose the neighbourhood to be all stars within a fixed bandwidth [9, 15, 17] (in our case pattern radius which is angular distance) or select the K-nearest neighbours as in [10,28]. Both approaches were tested. We used a fixed bandwidth as this enables more variation in local neighbourhood representations due to the varying star densities across the celestial sphere. This information is in itself a useful characteristic of a star-pattern. Whilst a fixed bandwidth could lead to star-patterns with few neighbours, the two-hop nature of our model allows nodes with few direct neighbours to aggregate information from further afield. The multiple attentions heads are able to learn the importance and relationship between node feature and edge attributes, making it possible to highlight nodes that are particularly reliable or informative for a target nodes prediction.

We filter the celestial graph by removing stars with a magnitude of 6 or higher (6.0 being the magnitude visible to the naked eye). We use a fixed pattern radius, τ, of 3 degrees to connect each star to its neighbours. Stars with less than 2 direct neighbours are also removed. With a 2-hop GNN, each star's receptive field can cover a radius of up to 6 degrees, and a diameter of up to 12 degrees.

The above method results in a graph of 4274 nodes and 20,310 edges spanning the night sky. This represents an idealised version of the celestial sphere under perfect conditions. In reality we will have to contend with various noise types that corrupt and diminish the information within an image, impacting our predictions. To ensure the models resilience to this noise we develop several functions to dynamically introduce noise into the celestial graph. Our GNN training is performed on a mixture of noisy variations, both in type and severity, to enable the resulting model to accurately predict star IDs from images captured under differing atmospheric and physical conditions. Whilst our celestial graph currently contains information about RA/DEC coordinates and absolute Vmag, this knowledge is only used to generate and add noise augmentation to the angular distance and relative magnitude data used during training and is not input to the model. This is because any test image, in a scenario with no apriori information, will be unable to determine this information. Instead angular distance between sources can be derived alongside a relative neighbourhood magnitude.

We consider real-world detection causes and effects to inform our choices for noise generation during training and tabulate the issue, the effect on output from source detection algorithms and how it is countered during training (Table 1). Where a problem is due to an algorithm inaccuracy, the sensor issue entry is left blank.

Positional Noise Positional noise refers to a shift in the observed position of a star/source. It is typically introduced during the source centroiding process as a result of atmospheric conditions, sensor noise, or optical aberrations such as lens distortion. In the context of our graph representation, it alters a star's connectivity and edge attributes (which rep-

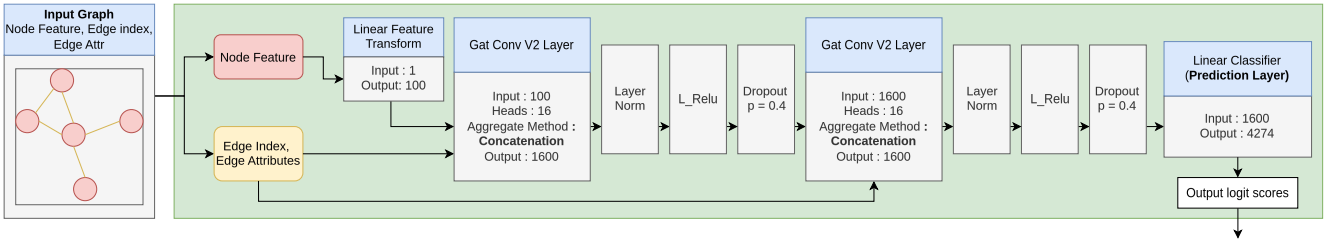

Figure 2. SIGNN model architecture for star pattern recognition using GAT layers.

Sensor issue	Source detection	Sample noise
Weather conditions (e.g. cloud cover)	Missing stars	Drop stars
	Incorrect magnitude	Magnitude noise
Object occlusion (e.g. branch)	Missing stars	Drop stars
Atmospheric scattering	Variation of star magnitude	Magnitude noise
Atmospheric scintillation	Variation of star position	Positional noise
	Variation of star magnitude	Magnitude noise
Incorrect centroid detection	Variation of star position	Positional noise
Lights (e.g. passing aircraft)	False stars	False star noise
Lens distortion	Variation of star position	Positional noise
Noisy CCD sensor (dark current)	Missing stars	Drop stars
	Incorrect magnitude	Magnitude noise
	False stars	False star noise

Table 1. Real-world detection issues are considered along with a suggested sample noise type during training.

resent the angular distance between stars). Existing work usually applies positional noise as a pixel offset on simulated images [28]. We instead apply it as an angular offset, directly altering the RA/DEC values to provide consistency with astrophysical data. This yields physically meaningful results independent of sensor properties. This approach enables objective comparison with known astronomical units, rather than unknown pixel quantities, ensuring reproducibility and precision in future error analyses and comparisons as discussed in the supplementary material.

Parameters detailed in Table 2 determine the range and intensity of noise. We use a randomly uniform distribution to determine the direction of each nodes offset (0-360°) and apply the angular distance following a Gaussian distribution. After applying this noise to all stars, we recalculate the angular distances between each node and rebuild the graph with the new data. Figure 3 shows an example.

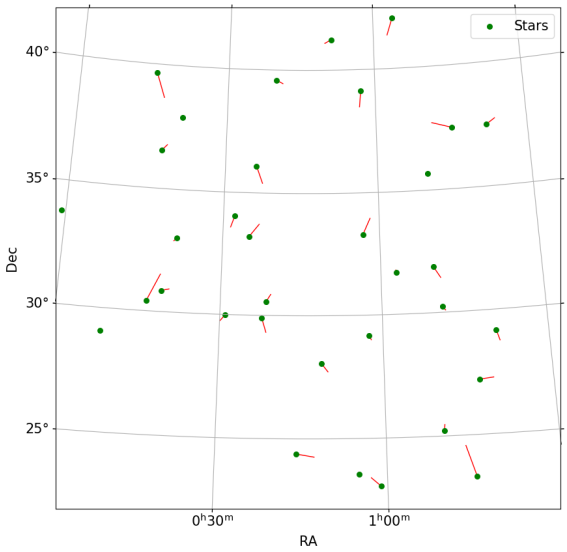

Figure 3. Example of positional noise, green markers show post-noise positions. Red tails indicate the shift from their original positions.

Dropped Star Noise This simulates missing stars due to atmospheric conditions, occlusion or noisy CCD sensors. Dropped stars distort real stars patterns as they reduce the connectivity of the graph and alter the calculated relative magnitudes. To simulate this, we randomly select a percentage of nodes within the graph to be dropped, removing them and their connections (see Figure 4)

False Star Noise This occurs during the source detection process, resulting in erroneous (false) sources, commonly caused by aircraft, sensor noise or satellites. False stars dis-

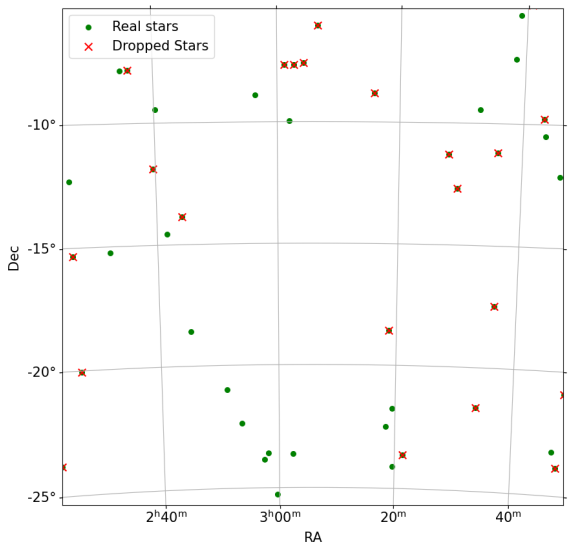

Figure 4. Example of dropped star noise. Stars marked with an x have been dropped from the graph.

tort real stars patterns by increasing the connectivity of the graph and altering the calculated relative magnitudes. Additionally, confident predictions on false stars and their subsequent use as a guiding star could lead to a breakdown in navigation. To simulate this noise, we randomly create a percentage of false nodes and insert them into the graph. Each false node's celestial co-ordinates and magnitude are randomly generated. Once complete we recalculate the angular distance and connectivity of all nodes (see Figure 5).

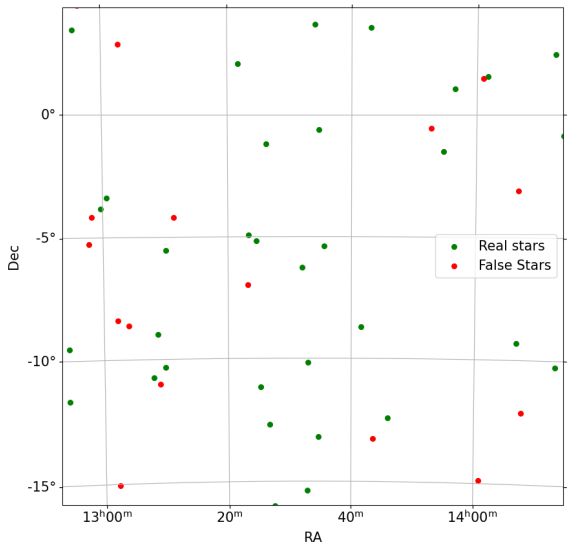

Figure 5. Example of false star noise, genuine stars are in green. False noise stars inserted into the graph are red.

Star Magnitude Noise This is a deviation in the perceived brightness of a star, caused by atmospheric conditions, light pollution or sensor noise. We apply magnitude noise as a randomly determined percentage modifier. Figure 6 shows an example of magnitude noise.

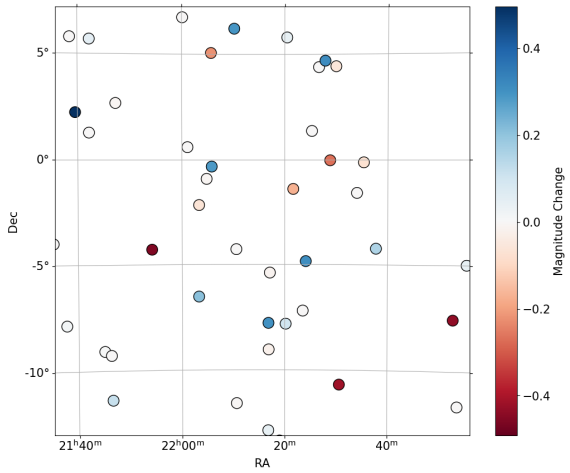

Figure 6. Example of magnitude noise, showing the resulting shift in each star's visible magnitude after noise is applied.

3.5. Sample Preparation

In real use cases calculating true magnitude values within an image is infeasible without calibration using already known stars. However, the relative magnitude of each star to their nearby neighbours can be obtained easily from the raw pixel brightness values. In our method, before a graph is sent to the model for training we first normalise (min-max scale) each stars magnitude relative to its directly connected neighbours. This effectively determines a local brightness ranking that is used for each node's feature and one that is capable of being calculated in a test image. This approach allows us to use magnitude and brightness as a meaningful feature in the identification process and is demonstrated to be successful in our results. Additional detail on brightness and relative magnitude is available in supplementary material.

3.6. Hyperparameters and Architecture

We used a Bayesian optimization process [2] to establish the optimal model structure and parameters of our GAT network, testing a variety of layer variations and parameters.

3.7. Training

We create a data-loader that generates variations of the initial celestial sphere graph each time it is accessed. This graph is stripped of each node's RA/DEC information before being sent to the model. The final graph prior to input contains nodes (whose feature is their relative magnitude),

an edge index (denoting connectivity), and edge attributes (angular distance between connected nodes). Each epoch generates 1024 variations with 20% sampled equally from each of the noise types and 20% with no noise. We train across 500 epochs using a batch size of 64 with a LR scheduler and early stopping enabled.

4. Results

4.1. Testing Using Simulated FOV Graphs

For testing we instead generate sub-graphs of the celestial sphere graph centred on randomly generated RA/DEC coordinates across the night sky, simulating all stars and connections in a 10° radius, which simulates a narrow-angle FOV image (FOV-graphs). Figure 7 shows an example FOV-graph generated by centering on Sirius, the brightest star in the night sky, illustrating it's node connectivity. Nodes beyond the outer ring are not simulated for this image, showing how the limited FOV clips the information available to the outer nodes. Nodes within the central ring are guaranteed to have their entire receptive field available within the image. This graph is passed as input to the model. We apply softmax to the resulting logits output to obtain each stars identity probabilities, the argmax of which is each nodes final prediction.

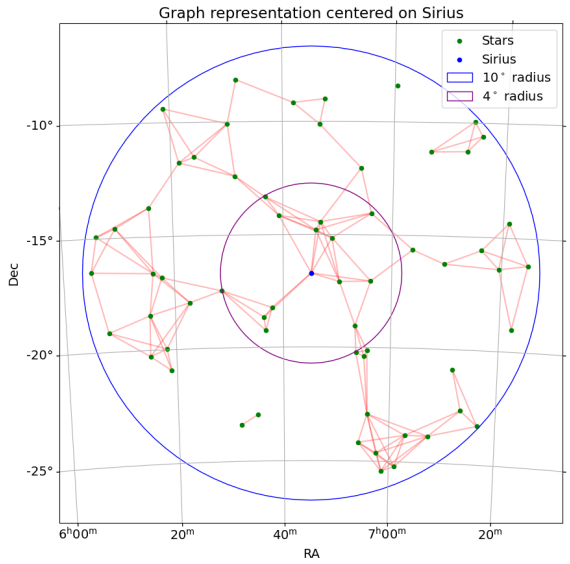

Figure 7. FOV-graph centered on Sirius with a 10° radius. Stars with an angular separation of less than 3° are connected to each other with an edge.

4.2. Testing Schema

Testing is structured to analyse each noise type at discrete noise levels. We generate randomly positioned FOV-graphs with specific noise types and intensity applied. We also test when all noise types are applied. We test 20,000 images at each stratified noise level as outlined in Table 2 resulting in a total testing of 800,000 simulated images encompassing a total of ≈ 28 million star predictions.

Noise Type	Start	End	Step
Position (Std Dev in radians)	0.0001	0.001	0.0001
Magnitude (Percentage)	1%	10%	1%
False (Percentage)	5%	50%	5%
Dropped (Percentage)	5%	50%	5%

Table 2. Testing Schema for Noisy Star Pattern Recognition. Each noise type is tested from the start to the end using the indicated step size.

4.3. Comparative Analysis

We compare with existing ML approaches [9, 26, 28] and classical approaches [15] by extracting, where possible, data from published results as described in the supplementary material. Table 3 outlines the main methodologies for each approach in their pattern construction and training process.

The comparator group all make predictions for a single star which is placed at the centre of the FOV. Our approach for SIGNN was to solve the more difficult problem of predicting all stars across the FOV simultaneously using the output logits for each node's star-identity. For comparison we show results for all predictions within 6°of the FOV centre. Since SIGNN outperforms the other methods using this more difficult approach, we did not test single star prediction which is an easier task and would achieve even better results for SIGNN.

Table 4 and Figure 8 shows each method's testing performance towards false noise (added stars). SIGNN outperforms comparative ML and classical approaches at each noise level, significantly so in the presence of higher percentages of false stars.

Whilst SIGNN is trained on drop noise as a percentage, to provide equivalent comparison we test using the same process as [9, 28] by dropping N stars.

SIGNN is shown in Table 5 and Figure 9 to outperform comparative ML and classical approaches at each noise level for dropped star noise.

As the comparator group use varying experimental setups and apply positional noise as a pixel deviation, which has a different meaning dependant on the simulated sensor, their results first need to be converted into a common reference frame of angular distance. SIGNN already trains and tests using angular distances. Details on the conversion process are available in supplementary material. Figure 10

Algorithm Training Comparison						
	Pattern Type	**Vmag**	**Target**	**Position Noise**	**False Noise**	**Dropped Noise**
SIGNN	All in radius	6.0	All stars in FOV	Degrees	Percentage	Percentage
RPNet [28]	N-Nearest	6.0	Single star	Pixels	Percentage	N Stars
Spider-Image CNN [9]	All in radius	6.0	Single star	Pixels	Percentage	N Stars
Grid Algorithm [15]	All in radius	6.0	Single star	Pixels	Percentage	N Stars

Table 3. The comparison group: Pattern type indicates whether nearest neighbour or a fixed pattern radius is used to define the neighbourhood. All algorithms are tested on starfields up to magnitude 6.0. Apart from our method, the approaches identify a single star at the centre of the FOV. Noise is added in each dimension as indicated, although we also test N dropped stars for fair comparison. Grid Algorithm parameters and results based on its use in RPNet [28].

Performance towards False Noise					
	10%	20%	30%	40%	50%
SIGNN	**0.989**	**0.963**	**0.935**	**0.903**	**0.866**
Spider-Image	0.987	0.883	0.702	0.531	0.267
RPNet	0.986	0.836	0.672	0.492	0.356
Grid Algorithm	0.933	0.837	0.750	0.608	0.303

Table 4. Performance for false stars. Additional stars were added to the starfield according to the indicated percentage. Results for RPNet [28], Spider-Image CNN [9] and Grid Algorithm [15] were obtained from [28], [9] and [28] respectively.

Performance towards N Dropped Stars					
	1	2	3	4	5
SIGNN	**0.993**	**0.987**	**0.981**	**0.975**	**0.968**
Spider-Image	0.946	0.899	0.777	0.600	0.390
RPNet	0.976	0.956	0.922	0.874	0.810

Table 5. Performance for N dropped stars. N stars were removed from the FOV and compared to results for RPNet [28] and Spider-Image CNN [9]. Drop results were unavailable in the literature for Grid Algorithm [15].

shows each model's performance in terms of angular distances. Whilst RPNet has a slightly higher initial accuracy, SIGNN outperforms all comparative methods as noise is increased, and is tested up to a larger positional movement.

Results for magnitude noise are discussed in supplementary material as its use and implementation varies significantly between SIGNN and the comparison group.

Confidence scores and filtering The impact of false stars on their surrounding predictions is visualized as an example in Figure 11. False stars are shown as red dots (•). Markers are shown for correct (dots) and incorrect (×) stars. These predicted stars are coloured by their confidence scores (viridis colour map). We observe incorrect predictions tend to have lower confidence.

Incorrect stars negatively impact navigation. To minimise this impact we propose using the confidence scores to filter the identifications and reduce the risk of selecting an incorrectly identified (or false) guide star. By threshold-

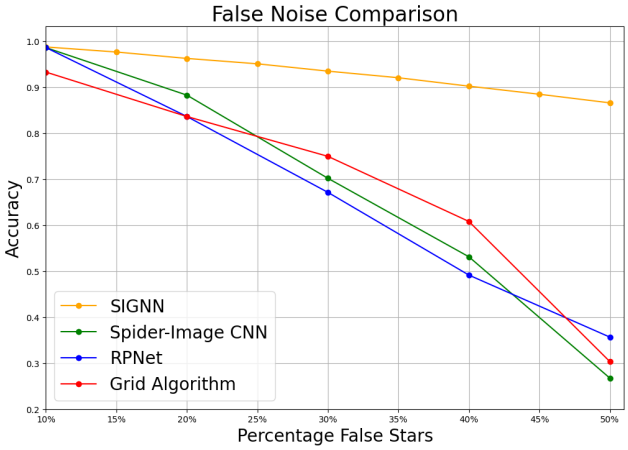

Figure 8. Graph representation of Table 4. False noise performance across noise levels compared with [9, 15, 28]. False noise from 10% to 50%.

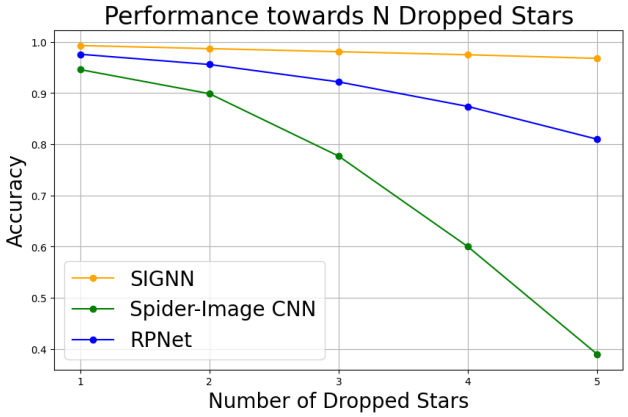

Figure 9. Graph representation of Table 5. Drop noise performance across noise levels compared with [9, 28]. Drop noise from 1 to 5 stars.

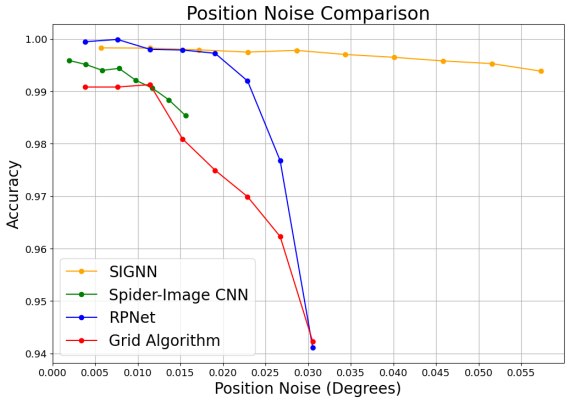

Figure 10. Position noise performance for standard deviation noise levels compared with [9, 15, 28].

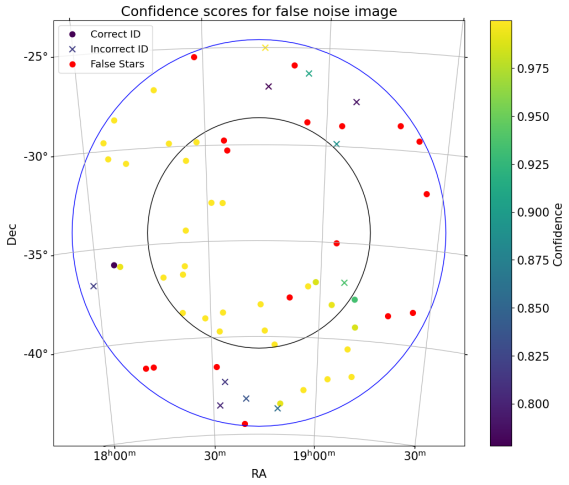

Figure 11. Subgraph showing correct/incorrect identifications coloured to prediction confidence scores. Markers for false stars are also visible. False noise of 28%.

Threshold	Correct Lost	Incorrect Lost
0.80	9.23%	92.11%
0.90	14.06%	97.20%
0.95	29.05%	98.40%
0.99	31.49%	99.68%
0.999	51.19%	99.99%

Table 6. Percentage of correct/incorrect predictions removed at different confidence thresholds.

ing results based on confidence we can remove the majority of incorrect results whilst retaining large portions of correct results as shown in Table 6. By filtering out predictions with a confidence lower than 0.999 we are left with less than five hundred incorrect IDs vs 13.6 million correct ones out of the original 28 million predictions.

All Noise We perform an additional test with all noise types applied simultaneously. We test 10 levels of noise, increasing each types intensity by 10% towards maximum value. Maximum value of positional noise is **0.001 radians**, dropped noise is **20%**, false noise is **20%** and magnitude noise is **4%**. Figure 12 shows how the model performs in this combined test.

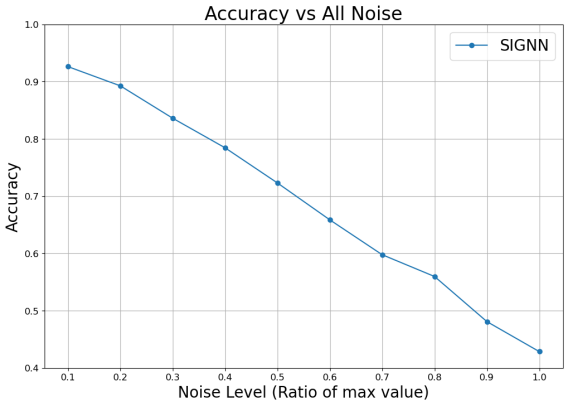

Figure 12. Accuracy for all noise types applied together.

5. Discussion and Future Work

SIGNN is shown to outperform existing classical and ML approaches to star identification, even when identifying all stars in an FOV (compared to existing approaches of identifying a single centrally placed star). We compare positional noise, commonly reported as pixel deviation, via angular distance offsets to allow a fair comparison between methods that are otherwise tied to a simulated sensor setup. SIGNN is initially trained and tested in this manner, allowing for easy comparison in future works and clearer interpretation of its performance on different sensors. SIGNN is modelled and evaluated at a threshold of 6.0 (for narrow-FOV sensors) and 5.0 (for wide-FOV sensors), available in supplementary material, showing it's broad applicability and ability to be trained towards varying use cases. Confidence thresholds are shown to be an effective and simple method to remove the majority of incorrect predictions.

Future work will explore using these scores to weight predictions, enabling cross-validation. Validated high confidence stars can be used to re-assess low confidence sources using the known catalogue. We aim to address the edge-of-image problem, where stars lose information when positioned towards the edge of an image. This would be achievable via a modified dataset and generation technique, able to represent this edge-of-image noise.

Acknowledgment Funded by EPSRC grant number EP/S021892/1. For the purpose of Open Access, the author has applied a CC BY license to any Author Accepted Manuscript (AAM) version arising from this submission.

References

[1] Coryn A. L. Bailer-Jones, Mike Irwin, and Ted von Hippel. Automated classification of stellar spectra - II. Two-dimensional classification with neural networks and principal components analysis. *MNRAS*, 298(2):361–377, Aug. 1998. 1

[2] Maximilian Balandat, Brian Karrer, Daniel R. Jiang, Samuel Daulton, Benjamin Letham, Andrew Gordon Wilson, and Eytan Bakshy. BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization. In *Advances in Neural Information Processing Systems 33*, 2020. 5

[3] Dor Bank, Noam Koenigstein, and Raja Giryes. Autoencoders, Apr. 2021. arXiv:2003.05991. 2

[4] Zelun Bao, Guiru Liu, Yefan Li, Yanxi Xie, Yang Xu, Zifeng Zhang, Qian Yin, and Xin Zheng. Pulsar identification based on generative adversarial network and residual network. *Complex Engineering Systems*, 2(4), 2022. 1

[5] Shaked Brody, Uri Alon, and Eran Yahav. How Attentive are Graph Attention Networks?, Jan. 2022. arXiv:2105.14491. 2

[6] Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A. Bharath. Generative adversarial networks: An overview. *IEEE Signal Processing Magazine*, 35(1):53–65, Jan. 2018. 1

[7] Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. *CoRR*, abs/1903.02428, 2019. 2

[8] Zhang Guangjun. *Star Identification : Methods, Techniques and Algorithms.* Springer, 2017. 1

[9] Jie Jiang, Lei Liu, and Guangjun Zhang. Star identification based on spider-web image and hierarchical cnn. *IEEE Transactions on Aerospace and Electronic Systems*, 56(4):3055–3062, 2020. 2, 3, 6, 7, 8

[10] C.C. Liebe. Star trackers for attitude determination. *IEEE Aerospace and Electronic Systems Magazine*, 10(6):10–16, 1995. 1, 3

[11] Daniele Mortari, Malak A. Samaan, Christian Bruccoleri, and John L. Junkins. The pyramid star identification technique. *NAVIGATION*, 51(3):171–183, 2004. 1

[12] Meng Na, Danian Zheng, and Peifa Jia. Modified grid algorithm for noisy all-sky autonomous star identification. *IEEE Transactions on Aerospace and Electronic Systems*, 45(2):516–522, 2009. 2

[13] Abderrahim Nabi, Zoubir Ahmed-Foitih, and Mohammed El-Amine Cheriet. Improved triangular-based star pattern recognition algorithm for low-cost star trackers. *Journal of King Saud University - Computer and Information Sciences*, 33(3):258–267, 2021. 1

[14] Keiron O'Shea and Ryan Nash. An Introduction to Convolutional Neural Networks, Dec. 2015. arXiv:1511.08458. 1

[15] C. Padgett and K. Kreutz-Delgado. A grid algorithm for autonomous star identification. *IEEE Transactions on Aerospace and Electronic Systems*, 33(1):202–213, 1997. 2, 3, 6, 7, 8

[16] Gustavo Pérez, Matteo Messa, Daniela Calzetti, Subhransu Maji, Dooseok E. Jung, Angela Adamo, and Mattia Sirressi. Starcnet: Machine learning for star cluster identification*. *The Astrophysical Journal*, 907(2):100, Feb. 2021. 1

[17] David Rijlaarsdam, Hamza Yous, Jonathan Byrne, Davide Oddenino, Gianluca Furano, and David Moloney. Efficient star identification using a neural network. *Sensors*, 20(13), 2020. 2, 3

[18] David Rijlaarsdam, Hamza Yous, Jonathan Byrne, Davide Oddenino, Gianluca Furano, and David Moloney. A survey of lost-in-space star identification algorithms since 2009. *Sensors*, 20(9), 2020. 1

[19] Snigdha Sen, Sonali Agarwal, Pavan Chakraborty, and Krishna Pratap Singh. Astronomical big data processing using machine learning: A comprehensive review. *Experimental Astronomy*, 53(1):1–43, Feb. 2022. 1

[20] Benjamin B. Spratling and Daniele Mortari. A survey on star identification algorithms. *Algorithms*, 2(1):93–107, 2009. 1

[21] Peter B. Stetson. DAOPHOT: A Computer Program for Crowded-Field Stellar Photometry. *Publications of the Astronomical Society of the Pacific*, 99:191, Mar. 1987. 1, 2

[22] M. C. Storrie-Lombardi, O. Lahav, Jr Sodré, L., and L. J. Storrie-Lombardi. Morphological Classification of galaxies by Artificial Neural Networks. *Monthly Notices of the Royal Astronomical Society*, 259(1):8P–12P, 11 1992. 1

[23] R.W.H. van Bezooijen. A star pattern recognition algorithm for autonomous attitude determination. *IFAC Proceedings Volumes*, 22(7):51–58, 1989. IFAC Symposium on Automatic Control in Aerospace, Tsukuba, Japan, 17-21 July 1989. 1

[24] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. *International Conference on Learning Representations*, 2018. accepted as poster. 2

[25] T. Vincenty. Direct and Inverse Solutions of Geodesics on the Ellipsoid with Application of Nested Equations. *Survey Review*, 23(176):88–93, Apr. 1975. 3

[26] Bendong Wang, Hao Wang, and Zhonghe Jin. An efficient and robust star identification algorithm based on neural networks. *Sensors*, 21(22), 2021. 2, 6

[27] Gangyi Wang, Jian Li, and Xinguo Wei. Star identification based on hash map. *IEEE Sensors Journal*, 18(4):1591–1599, 2018. 1

[28] Likai Xu, Jie Jiang, and Lei Liu. Rpnet: A representation learning-based star identification algorithm. *IEEE Access*, 7:92193–92202, 2019. 2, 3, 4, 6, 7, 8

[29] Guangjun Zhang, Xinguo Wei, and Jie Jiang. Full-sky autonomous star identification based on radial and cyclic features of star pattern. *Image and Vision Computing*, 26(7):891–897, 2008. 2

[30] Hongrui Zhao, Michael F. Lembeck, Adrian Zhuang, Riya Shah, and Jesse Wei. Real-time convolutional neural network-based star detection and centroiding method for cubesat star tracker. *CoRR*, abs/2404.19108, 2024. 1, 2

[31] Weiwei Zhao, Baoqiang Li, and Xiuyi Li. A star pattern recognition algorithm based on the radial companion-circumferential feature. *Mathematical Problems in Engineering*, 2022:1–10, 09 2022. 2

[32] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81, 2020. 1