Machine Learning for Exoplanet Detection: Identifying Exoplanets in Light Curves – RNN Component

Josh Manchester
University of Colorado Colorado Springs
jmanches@uccs.edu

Abstract

Josh Manchester — RNN Component (Revised Dataset Plan) Project: Machine Learning for Exoplanet Detection: Identifying Exoplanets in Light Curves Team: Tristan Moffett, Josh Manchester, Brianne Leatherman (UCCS, CS 4820: Artificial Intelligence, Dr. Adham Atyabi) I will build and test a Recurrent Neural Network (RNN) to spot exoplanet transits—small, regular dips in a star's brightness—in light-curve data. To keep scope realistic for the proposal phase, I will start with a small, curated subset that contains a handful of confirmed transit examples (e.g., ˜10 known planets) plus matched non-transit examples. If needed, I will add simple pseudo/synthetic data by injecting transit-shaped dips into real or cleaned light curves. I will try straightforward RNN variants (LSTM (a type of RNN (long short-term memory)) and GRU (a type of RNN (gated recurrent unit))) and report precision (share of predicted positives that are correct), recall (share of actual positives the model found), F1 score (precision (share of predicted positives that are correct) and recall (share of actual positives the model found)), and ROC-AUC (How well the model ranks positives over negatives across all thresholds). Results will be compared against a basic baseline and, later, the other team models (CNN/Transformer).

Introduction

Space telescopes record how a star's brightness changes over time. When a planet crosses in front of the star, the brightness dips slightly. Finding these small dips is hard because real signals can be weak and noisy (light traveling to us for years distorts it). RNNs are a natural fit for time-series and can learn patterns across many time steps. My goal is to train an RNN that can tell the difference between true transits and look-alike noise using a modest, well-labeled subset first, then expand if time allows.

Related Work

Exploratory sequence representation of Kepler light curves. Kügler et al. introduce an ESN-coupled autoencoder that encodes light curves via an RNN reservoir and optimizes reconstruction in sequence space (as opposed to readout space), highlighting the value of sequence-aware encoders for Kepler variability. This motivates using recurrent architectures to capture dynamics beyond pointwise features. RNNs for astrophysical transient detection in Kepler/TESS. Vida et al. evaluate LSTM-based flare detection and report ˜80–90% precision/recall at ¿5 on short-cadence Kepler and successful transfer to TESS, demonstrating that carefully trained LSTMs can generalize across missions and effectively reject false positives (e.g., RR Lyrae maxima). While their task is flares, their methodology informs my preprocessing, windowing, class imbalance handling, and thresholding choices. RNNs for event timing/intensity modeling. Du et al.'s RMTPP formalizes learning history-dependent intensity with RNNs for marked temporal point processes, showing how recurrent encoders capture complex event histories without brittle parametric assumptions. I will adapt the idea of encoding event histories (ingress/egress, gap structure) to augment classification with timing-aware auxiliary losses or features.

Datasets

Because the full missions are large, I will start small and focused: 1) Curated Subset (Primary): a small group of light curves with ˜10 confirmed transiting planets and a set of non-planet control stars. This keeps training/evaluation fast and easy to check by hand. 2) Pseudo/Synthetic Add-Ons (Optional): inject clean, transit-like dips into selected non-transit light curves to balance classes and stress-test the model. I will document any injection rules (depth, duration, period (time between repeats) jitter) so results are reproducible. 3) Expansion Path (Later): if time permits, scale up with more real examples or additional sectors/quarters.

Methodology

Preprocessing: simple detrending and normalization; careful handling of gaps; no label leakage (windows only use past/current data). Model: start with a 2–3 layer LSTM (also test GRU (a type of RNN (gated recurrent unit))) with dropout. Compare final-state (last hidden state) vs. time-pooled readout (output layer). Use a sigmoid (binary probability function)/softmax (multi-class probability function) head depending on sequence vs. window labels. Training: keep runs lightweight—small batch sizes, early stopping on validation F1. Handle class imbalance (far fewer

Dataset	Purpose	Notes/SOTA
Curated subset (∼10 known transits)	Train/val	Small, hand-checkable; class balance documented (TBD IDs).
Pseudo/synthetic injections	Stress tests	Inject depth/duration/period jitter; report injection policy.
(If used) Kepler/TESS shards	Generalization	Add a SOTA line when datasets are finalized (TBD).

Table 1: Planned datasets for the RNN component.

positives than negatives) with weights or focal loss (loss that down-weights easy examples). Apply light augmentation (noise/masking (blanking spans)) to mimic real conditions. Evaluation: report precision (share of predicted positives that are correct), recall (share of actual positives the model found), F1, AUC. Provide a short error analysis focusing on common false positives (stellar variability) and false negatives (very shallow or short transits). Optional timing helper (simple feature for period (time between repeats)/spacing): add a simple feature for expected transit spacing (period (time between repeats) guess) to give the RNN more context.

Experimental Plan & Milestones

Week 1: Assemble the small curated subset (˜10 positive, matched negatives) and write a minimal data-prep notebook. Week 2: Train a simple baseline (e.g., logistic regression on summary stats) to set a floor; implement the first LSTM (a type of RNN (long short-term memory)). Week 3: Tune window length and labeling; add GRU (a type of RNN (gated recurrent unit)); pick the better recurrent variant based on validation F1/AUC. Week 4: Add optional pseudo/synthetic injections to balance classes; re-evaluate and do a brief error analysis. Week 5–6: Polish: ablations (with/without timing helper (simple feature for period (time between repeats)/spacing)), final metrics/tables/plots, and a short write-up for the team paper.

Risks & Mitigations

• Too few positives: mitigate with pseudo/synthetic injections and careful cross-validation. • Overfitting the small set: use validation splits, early stopping, and simple models first. • Data cleaning surprises: keep preprocessing minimal and documented; track all changes in the notebook.

Conclusion

Starting with a small, well-labeled subset keeps the RNN work focused and feasible. If the initial results look good, I can scale up the dataset and compare my RNN to the team's CNN/Transformer models on the same evaluation plan.

References

[1] Kügler, S. D., Gianniotis, N., & Polsterer, K. L. (2016). An explorative approach for inspecting Kepler data. MNRAS, 455(4), 4399–4405.

[2] Vida, K., Bódi, A., Szklenár, T., & Seli, B. (2021). Finding flares in Kepler and TESS data with recurrent deep neural networks. A&A, 652, A107.

[3] Du, N., Dai, H., Trivedi, R., Upadhyay, U., Gomez-Rodriguez, M., & Song, L. (2016). Recurrent marked temporal point processes (model of when events happen): Embedding event history (past sequence of events) to vector. KDD 2016, 1555–1564.