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)

# Abstract

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 & Problem

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

This section reviews how recurrent neural networks (RNNs) have been applied to space-telescope light curves and what that means for my plan. I focus on three works: (1) An LSTM-based flare detector that operates on Kepler and TESS data (Vida et al., 2021), (2) An ESN–autoencoder representation learner designed for Kepler time series (Kügler, Gianniotis, & Polsterer, 2016), and (3) A recurrent marked temporal point process (RMTPP) model that shows how an RNN can encode event timing history (Du et al., 2016). Together these papers explain why sequence-aware models matter to what I am doing, and show that different RNNs tended to work well in practice, and how basic timing signals can help separate real periodic events, the transits, from look‑alike noise.

RNNs for light-curve event detection (Vida et al., 2021). Vida and colleagues trained and evaluated multiple RNNs for detecting stellar flares in Kepler and TESS photometry. Although flares are brightenings and transits are dimmings, both problems require learning patterns in long, noisy sequences. Their best-performing network stacked several long short-term memory layers (LSTMs), used dropout for regularization, or not memorizing noise, and applied a one‑unit sigmoid output for binary classification (Vida et al., 2021). The authors addressed class imbalance directly by weighting the minority class and tuned hyperparameters over the entire system. An important insight was that including “hard negatives” (non‑flare astrophysical signals) during training reduced false positives and turned out to be very practical in use. Most relevant for our project, a model trained on Kepler generalized well to TESS even though the cadence and noise profiles differ. That result supports my plan to keep preprocessing consistent across sources, use stacked LSTMs or GRUs first, and report precision, recall, F1, and ROC–AUC in a way that is holds to threshold choice (Vida et al., 2021).

Two details from Vida et al. (2021) feed directly into my setup. First, the authors compared gated recurrent unit (GRU) and LSTM variants and found that stacked LSTMs handled “astrophysical noise” better in their flare task. With that knowledge, I will start with a 2–3 layer LSTM (128–256 units) and keep a GRU as a secondary baseline. Second, they used strong, simple regularization techniques, that is dropout and early stopping, and balanced the classes explicitly. Because transit positives will be rare in my small subset (≈10 known systems), I will go along the same lines and also try focal loss as a variant when class weighting might not be enough to control any false positives (Vida et al., 2021).

Sequence-aware representation with ESN–autoencoders (Kügler et al., 2016). While Vida et al. focus on supervised detection, Kügler and colleagues show how to learn lower‑dimensional representations of Kepler light curves that still respect sequence structure. They pair an Echo State Network (ESN) with an autoencoder: the ESN maps each light curve to a set of weighted readouts, and the autoencoder compresses and reconstructs those weights. Their key difference is to evaluate the reconstructions in the original sequence space by reinserting the reconstructed weights back into the ESN to get a reconstructed light curve; the loss is computed on that sequence rather than only on the readout (Kügler et al., 2016). This is an important change because it forces the model to preserve time‑dependent behavior.

For my work, the ESN–autoencoder paper provides two practical observations. First, any diagnostics I use should look at the whole sequence, not just a final embedding. I will need to use evaluation metrics on windows and on full sequences wherever possible and include plots that compare the original and model‑highlighted spans of time from the data. Second, if I include any representation learning step (e.g., pre‑training or a feature extractor), I need to prefer objectives that score performance in the sequence space. Even though I am not actually building an ESN–autoencoder, aligning training and evaluation with the sequence will help me avoid shortcuts and can help me through my error analysis (Kügler et al., 2016).

Timing-aware RNNs via RMTPP (Du et al., 2016). Du and colleagues introduce RMTPP, which uses an RNN to search out event histories and rewrite the data as a temporal point process. Rather than predicting only labels, the model can start to guess or predict when the next event is likely to occur based on the sequence of data we have processed so far. In other words, RMTPP turns the RNN into a history based dependent source and intensity model that, together, handles event timing and event type (Du et al., 2016). Although RMTPP was demonstrated other areas like finance and healthcare, the idea can be directly applied to periodic transit detection: a light curve with a planet has a regular rhythm (ingress to egress, repeat), while many false positives are random in periodicity or quasi‑periodic. A light touch of timing information can help the classifier.

Knowing this, I will attempt to adapt RMTPP’s spirit in a lightweight way. I am not replacing the classifier with a full point‑process model, but I will attempt to add a simple timing helper that captures intervals between candidate transit edges or captures data in a rough period guess when available. This “helper” might be an auxiliary feature or a small auxiliary loss that pushes for consistency within a periodic framework. The goal is to help the RNN learn that true transit signals repeat on a calculatable schedule, which should reduce confusion with isolated dips or irregular stellar dips in beightness (Du et al., 2016).

Putting the three paper’s ideas together. Vida et al. (2021) show that stacked LSTMs trained with class weighting and clean preprocessing can detect rare events in long, noisy sequences and even transfer across different photogrammetric space missions; I will try to mirror that recipe for greatest success. Kügler et al. (2016) show that we need to respect full‑sequence structure when learning representations or doing diagnostics; for this I will try to evaluate based on sequences/windows and picture the model attention over time. Du et al. (2016) have given me a blueprint for bringing timing into the loop; I will attempt to add a light timing signal to nudge the RNN toward known or apparent periodic dips in the light curves without complicating the data being feed/read. These choices should be simple and reproducible at the outlook, and they should fit the scope of a small, curated dataset I plan to use while still setting up a fair comparison against my team’s CNN/Transformer models.

Planned evaluation and error analysis based on related work. Following Vida et al. (2021), I will attempt to report precision, recall, F1, and ROC–AUC and examine confusion matrices for common failures that may or will occur. False positives should be grouped together by likely source (e.g., variable stars, systematics, residual trends), and false negatives will be checked for very shallow or short transits which can mean something else caused the dip in the light. From Kügler et al. (2016), I am borrowing the idea of sequence‑level diagnostics which means I will be plotting model scores across time to ensure the network’s focus lines up with actual transit starts and stops. From Du et al. (2016), I will evaluate whether the timing “helper” reduces false positives due to variability in the ingress and egress period, without hurting recall on genuine periodic signals. Together, these checks turn my related work into solid testable steps.

Implications for dataset size and generalization. Vida et al. (2021) showed that consistent preprocessing and using a balanced set of training data matter more than other architectures for the best performance; that should be good news for me since I am using a small curated subset. If my results are promising, I should be able to scale my RNN to look at more stars or additional sectors/quarters and repeat the same evaluation plan. In the future, I plan to track whether the timing helper still “helps”, and whether stacked LSTMs remain better than GRUs, paralleling the comparisons in Vida et al. (2021). Throughout my testing I will keep in mind that sequence‑based diagnostics are a key to performing my transit search in the data, from Kügler et al. (2016) so that changes in the data are grounded in what I can dig out of the time‑series behavior.

Summary. The RNN literature relevant to light curves argues for (a) stacked LSTMs or GRUs with explicit class‑imbalance handling and conservative regularization (Vida et al., 2021), (b) representation and diagnostics that respect sequence structure (Kügler et al., 2016), and (c) simple timing cues that help the network prefer periodic, transit‑like patterns (Du et al., 2016). I will carry these into my RNN baseline so that it is both technically sound and scoped for our project timeline.

# Datasets — Initial Plan

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.

|  |  |  |
| --- | --- | --- |
| Dataset | Purpose | Notes on SOTA/benchmarks |
| Curated subset (~10 known transits) | Train/val | Small, hand-checkable; class balance documented (TBD exact IDs). |
| Pseudo/synthetic injections | Stress tests | Injected transit depth/duration/period jitter; report injection policy. |
| (If used) Kepler/TESS shards | Generalization | Cite a relevant SOTA line once datasets are locked (TBD). |

# Methodology (RNN)

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 called 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 positives than negatives) with weights or focal loss (loss that down-weights easy examples). Apply light augmentation (noise/masking or 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 guesses) to give the RNN more context.

# Experimental Plan & Milestones (Josh)

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 that uses 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 which is a simple feature for period or dip 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.

# Short 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

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

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.

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.