# Machine learning the visible counterparts to gravitational wave events: kilonovae

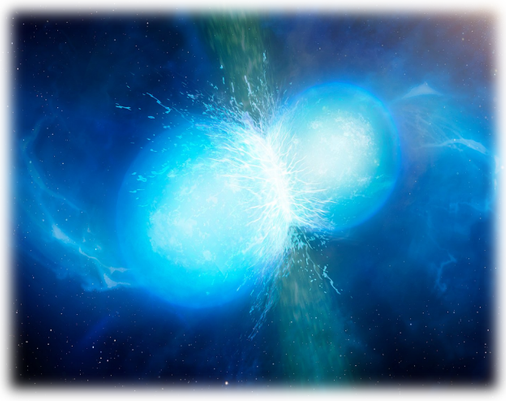
**Utkarsh V. Mali[1], Keir K. Rogers[2], Mattia Bulla[3]**

[1]Department of Physics, University of Toronto
[2]Dunlap Institute, University of Toronto
[3]Department of Astronomy, Stockholm University

UofT Astro

Fig 1 (right): Binary neutron star merger.

**Kilonovae are the visible emissions from BNS and NS-BH mergers.**



**Radiative transfer simulations: ground truth**

**Dimensional reduction (PCA)**

**Machine learning model training**

**Predict kilonova fluxes**

**Parameter inference**

**Introduction**
Kilonovae are transient astronomical events that should occur during binary neutron star (BNS) or neutron star black hole (NS-BH) mergers. Electromagnetic radiation is emitted due to the rapid neutron capture process (r-process) from the material ejected from the merger. This neutron capture is responsible for the formation of heavy elements such as gold, uranium etc. Kilonovae are a new class of transients, studying them may give rise to a better understanding of the Universe. When used as a counterpart to the gravitational waves, kilonovae can constrain properties of the merger, breaking degeneracies in standard siren measurements [1, 2].

**Radiative transfer simulations are too computationally expensive and may take weeks to be used in parameter inference. ➔ ML Alternative (Gaussian Process)**

We offer a solution using machine learning that offers a huge runtime speed-up. Our model is a Gaussian processes (GP) emulator [3]. This was chosen for its ability to compute quickly and to propagate errors through the computation.

**Methodology**
The pipeline begins with organizing the ground truths of the Bulla model by kilonovae parameters in dimensionality; viewing angle, dynamical ejecta mass, wind ejecta mass and half-opening angle. These data are reduced along the time-wavelength domain using principal component analysis (PCA) [4]. The GP emulator uses the principal components from this transformation as the training vector Y. The kilonovae parameters are used as the input vector X. Once trained the emulator predicts a set of PCA components which are converted into fluxes and ultimately photometric band magnitudes. Parameter inference occurs using MCMC sampling [5]. From the user's point of view, they input any set of light curve magnitudes. Then the software will provide an estimation of the parameters posterior distribution associated with that kilonova event.

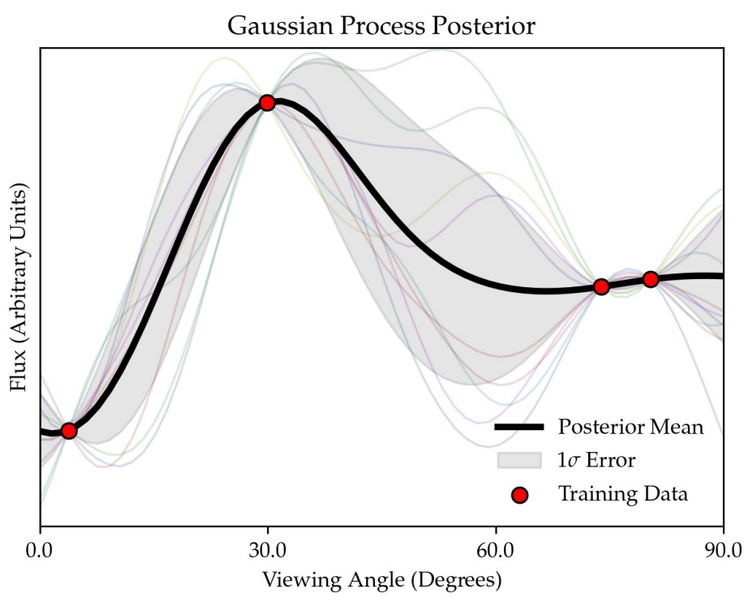
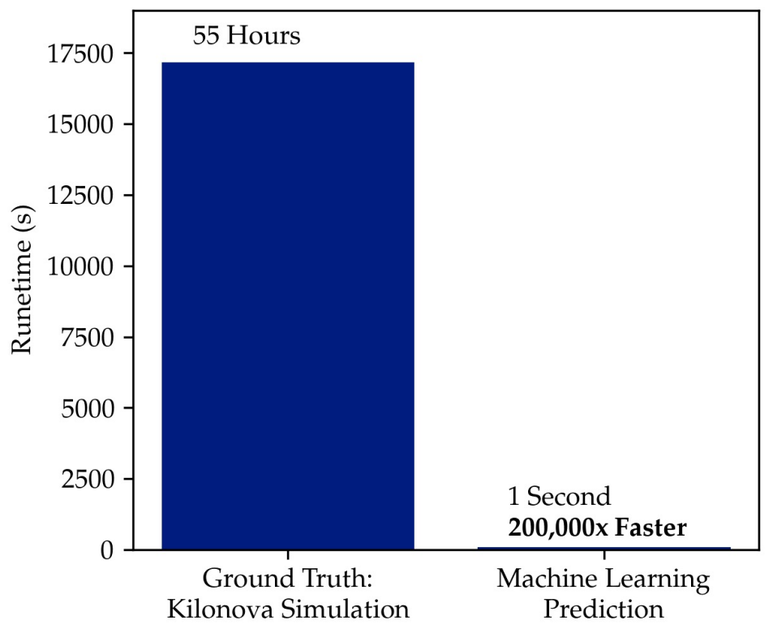



Fig 2 (Left): Example Gaussian process. Random draws from an optimized GP are shown in color with its posterior mean shown in black. The errors of the posterior are lowest near the training data and increase away from them.

**Results and Limitations**
The emulator takes around 45 to 60 CPU-seconds to train and 1 CPU-second to predict. This is a huge computational speedup compared to the original data's 50–55 CPU-hours runtime. Parallel computing was implemented in order to improve further on the computational runtime. This was done by saving training data in partial chunks and simultaneously training multiple emulators. As seen in Figure 4, the model shows strong performance overall. The average fractional error between the radiative transfer model and the machine learning prediction is 0.1. There is a 68% ($1\sigma$) chance that the fractional error will be below 0.06 and a 95% chance ($2\sigma$) that the fractional error will be below 0.4. While the emulator shows good overall performance, it has shown to have worse error during the first 3 days post-merger, the average fractional error during this period is 0.8 while the average fractional error for the remainder is 0.05. This was due to larger early variation of flux in the radiative transfer simulations, a simple solution to this would be to use multiple emulators on different parts of the dataset. Finally, the emulator's use of PCA to recover the fluxes made it non-trivial to propagate error from the GP to photometric band magnitudes.

Fig 3 (Right): Computational speed up using machine learning over radiative transfer simulations.



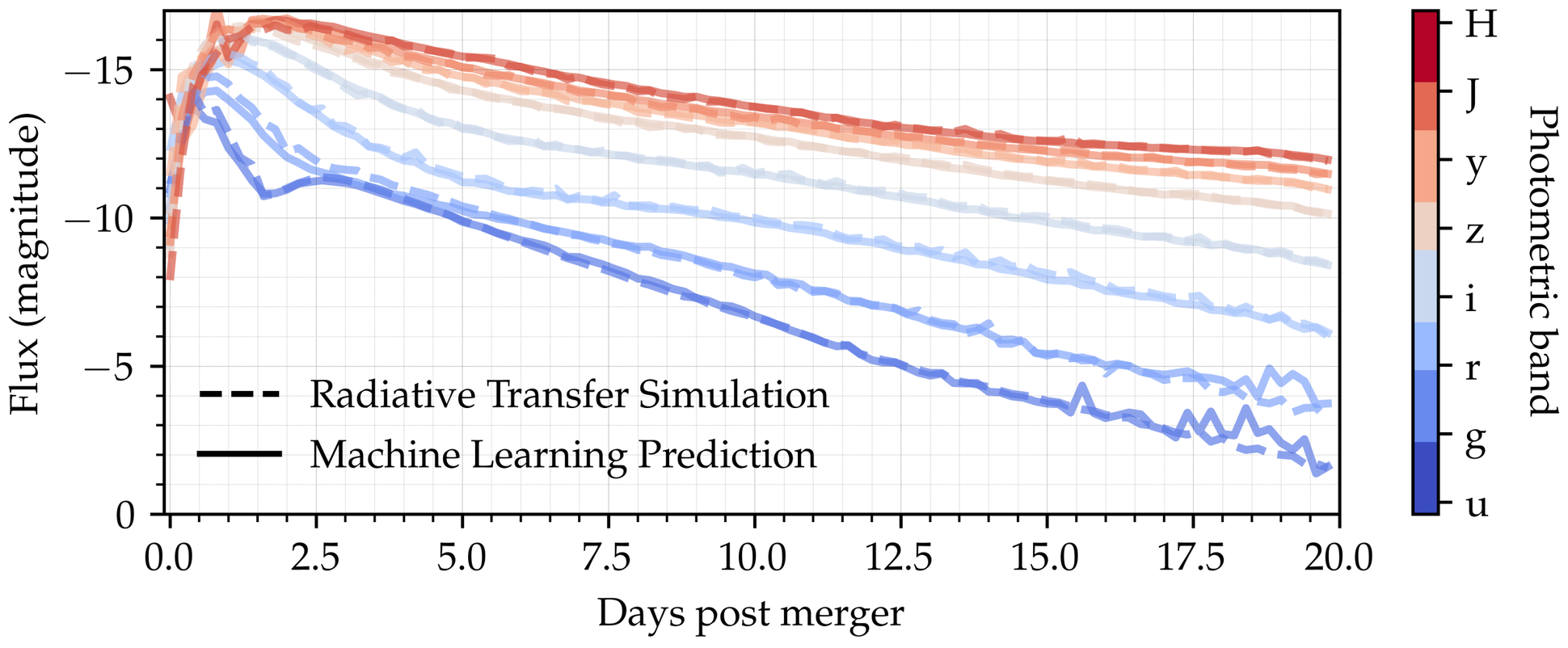




Fig 4 (Above): A sample simulation of kilonova magnitude as a function of time. Eight photometric bands are shown with their respective ground truth. There is good agreement between truth and prediction.

**Future Progress**
The final stage of the pipeline uses MCMC to infer parameters of a new set of kilonova data. The algorithm uses a uniform prior and Gaussian likelihood. This stage of the project is still ongoing. We are working with multiple groups who see potential uses, for example, using the emulator to constrain parameters in the UV band range. There is also possibility to expand this emulator to train on multiple different radiative transfer models which will reduce its reliance on the assumptions of a single model. Ultimately, this software can be used to parametrize light curves into known elements of the kilonovae.

**Open Source @ https://git.io/JRs76**

**Acknowledgements**
I would like to thank the Summer Undergraduate Research Program for allowing me the opportunity and funding to conduct research in astrophysics, my supervisor Dr. Keir Rogers for sharing his wisdom and experience about academia and research and all the other mentors, such as Professor Renee Hlozek that have guided me throughout my time with the department.

**References**
[1] Coughlin, Michael W., et al. "Standardizing kilonovae and their use as standard candles to measure the Hubble constant." *Physical Review Research* 2.2 (2020): 022006.
[2] Bulla, Mattia. "POSSIS: predicting spectra, light curves, and polarization for multidimensional models of supernovae and kilonovae." *Monthly Notices of the Royal Astronomical Society* 489.4 (2019): 5037-5045.
[3] GPy [Computer software]. Retrieved from https://github.com/SheffieldML/GPy
[4] Coughlin, Michael W., et al. "Constraints on the neutron star equation of state from AT2017gfo using radiative transfer simulations." *Monthly Notices of the Royal Astronomical Society* 480.3 (2018): 3871-3878.
[5] Emcee [Computer software]. Retrieved from https://github.com/dfm/emcee