Summer 2025 Research Summary
Machine Learning Flags Fast Neutrino-Flavor Instabilities

John McGuigan Advisor: Sherwood Richers
Department of Physics and Astronomy, University of Tennessee, Knoxville

Abstract. Core-collapse supernovae and neutron star mergers host extreme conditions where neutrino self-interactions can trigger *fast flavor instabilities* (FFI) on cm/ns scales. I trained heavily regularized multi-layer neural networks (MLNNs; 4–6 layers, hundreds of neurons wide) in PyTorch on $\sim 8 \times 10^5$ labeled snapshots spanning seeded-unstable, guaranteed-stable, and neutron-star-merger (NSM) cases. After a systematic hyperparameter search (dropout, batch normalization, weight decay, batch size, learning rate schedules, and early stopping) and class-weighted training to address the ∼5–6% unstable minority class, the best model reaches $F_1 \approx 0.95$ on held-out data while remaining small enough for in-situ use in simulation codes. The network outputs a calibrated $P(\text{FFI} \mid x)$ per grid cell, replacing an expensive dispersion solve with negligible overhead in principle; integration tests and wall-time benchmarks are planned for production hydrodynamics runs.

Background & Physics

Neutrinos carry away $\gtrsim 99\%$ of the gravitational binding energy in core collapse, and their angular electron-lepton-number (ELN) spectra can admit crossings that seed FFI. In the mean-field limit, the flavor density matrix $\rho_{\mathbf{p}}$ is

$$i\,\partial_t \rho_{\mathbf{p}} = [H_{\text{vac}} + H_{\text{mat}} + H_{\nu\nu},\, \rho_{\mathbf{p}}]\,,$$

with the self-interaction term

$$H_{\nu\nu} \propto \mu \int (1 - \cos\theta)\, G(\mathbf{v})\, d\Omega,$$

where $G(\mathbf{v})$ is the ELN angular distribution and μ sets the interaction scale. The presence of ELN crossings can drive flavor conversion on microscopic (cm/ns) scales that are numerically difficult to resolve directly across all cells in neutrino transport simulations.

Data & Features

Labeled examples consist of $\sim 8 \times 10^5$ per-cell snapshots (stable versus FFI-unstable) drawn from CCSN and NSM datasets. The feature vector holds 27 scalars per cell capturing thermodynamic and neutrino-moment quantities such as density, Y_e, entropy, flux integrals, and ELN-crossing proxies. The dataset is imbalanced with only ≈5–6% unstable cells, and an 80/20 train/validation split with fixed folds and multiple deterministic seeds tests reproducibility.

Model & Training

The classifier is a fully connected MLP with 4–6 hidden layers a few hundred neurons wide, using ReLU activations, BatchNorm, and dropout. Training employs class-weighted binary cross-entropy with L_2 regularization,

$$\mathcal{L} = -w_1\, y \log \hat{y} - w_0 (1 - y) \log(1 - \hat{y}) + \lambda \sum_{\ell} \|W_\ell\|_2^2,$$

and is optimized with AdamW and a cosine/step learning-rate schedule. Batch sizes of 4 096–16 384 start near 3×10^{-3} learning rate with decay and early stopping to prevent overfitting. After training, the classification cutoff on $P(\text{FFI} \mid x)$ is swept to maximize the F_1 score, balancing precision and recall.

Validation & Reproducibility

Precision, recall, and $F_1 = \frac{2PR}{P+R}$ are measured on a held-out test set. Results at the optimal cutoff remain stable across seeds; representative numbers are below.

Seed	Recall	Precision	F_1
17	0.921	0.983	0.951
43	0.919	0.974	0.945

Median performance across top configurations sits near $F_1 \approx 0.94$–0.95 with high precision ($\approx$0.97–0.98) and strong recall ($\approx$0.90–0.92). I also trained smaller variants (fewer layers and narrower widths) to minimize inference cost; these retain competitive F_1 with modest drops in recall.

Deployment Plan & Impact

For in-situ prediction, the model is exported to TorchScript and run as batch per-cell inference on CPU or GPU within radiation-hydrodynamics codes. The resulting $P(\text{FFI} \mid x)$ serves as a flag or risk score that enables flavor-aware closures or triggers higher-fidelity solvers only where instability is likely. The architecture is constrained for real-time deployability, and end-to-end wall-time benchmarks on production grids are the next step.

Progress to Date

I built an end-to-end PyTorch training pipeline with data loaders, augmentation hooks, logging, and plotting. Automated hyperparameter searches span depth, width, dropout, weight decay, batch size, and learning-rate schedules with early stopping. Class-weighted training handles severe imbalance, and threshold sweeps with calibration curves set operating points. Reproducibility studies across multiple seeds provide convergence diagnostics and overfitting detection. I also developed compact models that retain high sensitivity to the minority class while enabling faster inference.

Next Steps. Runtime profiling inside a test hydrodynamics code; uncertainty calibration (temperature scaling, isotonic regression), out-of-distribution checks for new progenitors/NSM conditions, and active-learning loops that prioritize follow-up dispersion solves where the classifier is uncertain.