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 $\rho_{\mathbf{p}}$ obeys

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

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 prohibitive to resolve directly across all cells in multidimensional simulations.

Data & Features

- Labeled examples: $\sim 8 \times 10^5$ per-cell snapshots (stable vs. FFI-unstable), drawn from CCSN and NSM datasets.
- Feature vector: 27 scalars per cell (thermodynamic and neutrino-moment quantities including density, Y_e, entropy, flux integrals, ELN-crossing proxies).
- Class balance: ≈5–6% unstable.
- Train/validation split: 80/20 with fixed folds; multiple deterministic seeds used to test reproducibility.

Model & Training

- Architecture: fully-connected MLP (4–6 hidden layers, widths in the few-hundreds), ReLU activations, BatchNorm, dropout.
- Loss & imbalance handling: class-weighted binary cross-entropy with L_2 regularization,

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

optimized with AdamW and a cosine/step learning-rate schedule.

- Batching & schedules: batch sizes 4 096–16 384; learning rate $\sim 3 \times 10^{-3}$ with decay; early stopping to prevent overfitting.
- Thresholding: sweep the classification cutoff on $P(\mathrm{FFI} \mid x)$ to maximize the F_1 score (balances precision and recall).

Validation & Reproducibility

- Metrics: precision, recall, $F_1 = \frac{2PR}{P+R}$ on a held-out test set.
- Results at the best cutoff are 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 for in-situ usage; these retain competitive F_1 with modest drops in recall.

Deployment Plan & Impact

- **In-situ prediction:** export to TorchScript/ONNX and batch per-cell inference on CPU or GPU within radiation-hydrodynamics codes.
- **Usage:** treat $P(\mathrm{FFI} \mid x)$ as a flag or risk score to (i) enable flavor-aware closures, or (ii) trigger higher-fidelity solvers only where the network predicts instability.
- **Speed-aware design:** architecture constrained for real-time deployability; end-to-end wall-time benchmarks on production grids are the next step.

What I Build So Far

- End-to-end PyTorch training pipeline (data loaders, augmentation hooks, logging, and plotting).
- Automated hyperparameter searches over depth, width, dropout, weight decay, batch size, and learning-rate schedules with early stopping.
- Class-weighted training to handle severe imbalance; threshold sweeps and calibration curves to set operating points.
- Reproducibility studies across multiple seeds; convergence diagnostics and overfitting detection.
- Compact models for faster inference while maintaining high sensitivity to the minority class.

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.