AwareGPT: Detecting Hallucinations in Foundation Models

Manuel Cardenas, Carl Shikang Liu, Austin Chen {manuelcp, csliu, austin05}@stanford.edu

Project Overview

- Local, open-source LLMs have the potential to democratize knowledge in communities without internet (think of a local tutor). **But they hallucinate.**
- ***Can we predict the hallucinations of a small, local model before they happen?***
- We trained classifiers on model internals

Prior Work

- Different approaches have been tried to predict hallucinations. We filtered them on which would be feasible on a low-spec android phone
- We follow Snyder's (2024) approach: training classifiers on the model internals (classifiers on fully connected layers and attention layers)
- **Benchmark: AUROC of 0.8**

Future Work

- Fine-tune the model to give itself an understanding of what it does not know.
- Broaden the definition of hallucination. In a sentence, some things might be right and some might be hallucinated
  - Quantify factuality through fact decomposition
- Integrate with kiwix.org to improve scores

iOS App

We built ConfidentLM, an app that uses our trained NN and shows how confident the model is in each of its responses

Methods

We "interviewed" Liquid AI's LFM2-1.2B to gather two datasets (~20k rows, ~70% hallucinated, labelled as 1):

Dataset 1: Deep Internal Features ("White Box")
- Used unoptimized "eager" execution to capture model internals.
- Extracted the hidden state outputs of each transformer layer (17x2048) and Attention Maps 6x32x25x25 per generation.

Dataset 2: Deployment Features ("Black Box")
- Used quantization and FastAttention to simulate restricted "edge" environments (e.g., mobile).
- Limited to the final embedding (1x2048), logits (1x65536), softmax probabilities (1x65536), and token logprobs (1x100).

We trained a series of classifiers. The best performing were:

- **Random Forest:** Uses 200 parallel decision trees (max depth 20) using Gini impurity. We utilized balanced class weights to inversely weight classes (1 = hallucination is majority class)
- **Gradient Boosting:** A sequential ensemble of **100 shallow trees** (depth 5). It minimizes Log Loss via gradient descent with a **learning rate of 0.1**, iteratively correcting errors from previous trees.
- **Neural Network (MLP):** A 4-layer feedforward network compressing features from input → 512 → 256 → 128 → 64 → output.Utilized **Batch Normalization** and **Dropout** (0.2–0.4) to prevent overfitting on high-dimensional embeddings.

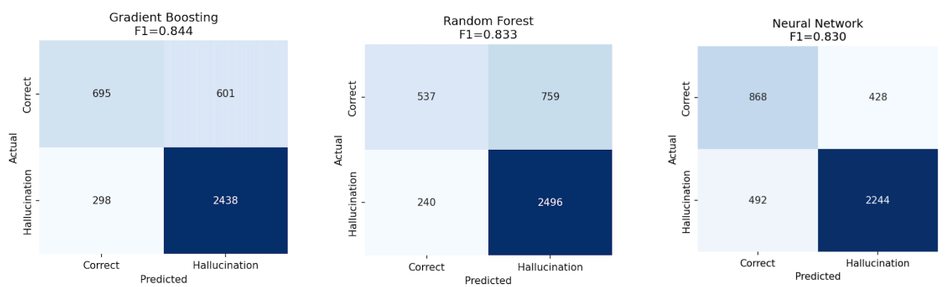
Results

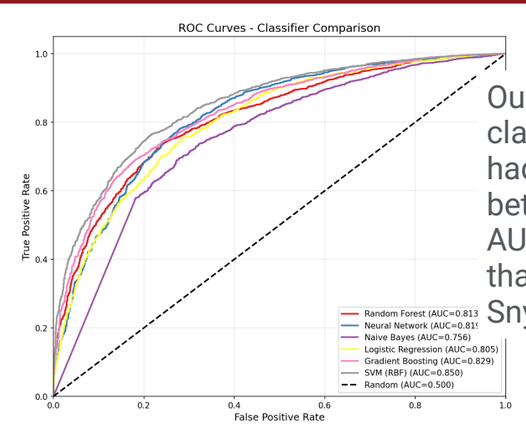
- Classifiers trained on dataset 1 reached a similar performance as Snyder.

Feature Method	Model	F1 Score	AUROC	Precision	Recall	Accuracy
Manual Stats	NN	0.789	0.500	0.651	1.000	0.651
	RF	0.806	0.738	0.710	0.932	0.708
Attention PCA	NN	0.737	0.751	0.796	0.687	0.687
	RF	0.797	0.710	0.691	0.941	0.692

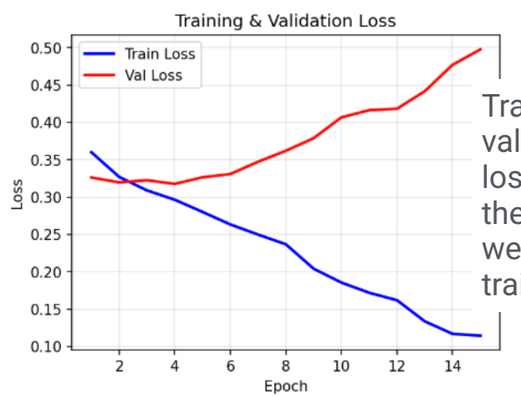
- Classifiers trained on dataset 2 **performed even better** (less features).

Model	Accuracy	Precision	Recall	F1 Score	AUROC
Gradient Boosting	0.777	0.802	0.891	**0.844**	0.829
Random Forest	0.752	0.767	**0.912**	0.833	0.813
SVM (RBF)	**0.779**	**0.861**	0.804	0.831	**0.850**
Neural Network	0.772	0.840	0.820	0.830	0.827





Our classifiers had a better AUROC than Snyder's



Train and validation loss of the NN we trained

[1] K. Clark *et al.*, 2019 [2] K. Cobbe *et al.*, 2021 [3] T. Dao *et al.*, 2022 [4] S. Farquhar *et al.*, 2024 [5] M. Geva *et al.*, 2020 [6] D. Hendrycks *et al.*, 2021 [7] M. Joshi *et al.*, 2017 [8] Liquid AI, 2025 [9] J. Luo *et al.*, 2024 [10] N. Nonkes *et al.*, 2024 [11] E. Quevedo *et al.*, 2024 [12] B. Snyder *et al.*, 2024