ECE/CS/ME 539 Introduction to Artificial Neural Networks

Project Progress Report: Tricking AI Detectors

Team #21
Joseph Skantz - Undergraduate
Malachi Clark - Undergraduate
Ayush Sharma - Undergraduate
Date of submission: May 1, 2025

Abstract

Throughout the course of our project, Tricking AI Detectors, several key findings have been made. Our binary classification when implemented through a logistic regression model was trained to convergence with an accuracy of 82.67%. Additionally, our binary classification, implemented through a fully connected neural network, has been trained to convergence with about 0.25 loss, and an accuracy of 91.34%. We saw a 2% increase in “human likeness” score as compared to the LLM before fine tuning. Although our LLM results were not what we wanted them to be, they were an indication in the right direction and have proved we can improve these results given more time and computational resources.

Introduction:

a. Overview
    i. The goal of this project is to develop an AI model capable of fooling AI text detection algorithms and programs. By training a binary classification model on a dataset that differentiates between human and AI text, we aim to predict text as human or AI while also generating AI-based human-like text, using verbosity tools like perplexity and entropy analysis to fine-tune the model.
b. Motivations & Significance
    i. What makes this project particularly compelling to us is its direct relevance to academic settings, where AI text detection tools are increasingly being deployed to evaluate student work. With the rise of large language models (LLMs), students may either intentionally or unintentionally rely on AI-generated content, prompting institutions to employ detection models as a means of maintaining academic integrity. However, we are interested in exploring whether these classification models can be challenged or manipulated, particularly when AI-generated text is strategically designed to imitate human writing convincingly. If such models can be cast into doubt due to the subtle yet significant distinctions between genuine human writing and AI-crafted text optimized for human-like qualities, it raises important questions about their reliability and fairness in high-stakes environments like education.
c. Related works
    i. During our research, we encountered a study[1] that sought to evaluate the effectiveness of software designed to detect AI-generated writing. The study specifically examined the performance of 16 different AI text detection models by comparing their ability to distinguish between 42 student-written essays and 42 AI-generated essays. The results from this study sparked our curiosity, particularly regarding the feasibility of misleading these detectors by using AI-generated text that closely mimics human writing. This raised an intriguing question: Could AI-generated text, when carefully optimized to appear more human-like, successfully evade detection by these models?
    ii. Additionally, we came across another study[2] that explored the inherent complexities of identifying machine-generated text. This study emphasized that the process of determining whether a piece of text was written by a human or an AI is far more intricate than a simple binary classification. Rather than a clear-cut distinction, the study underscored the nuanced nature of textual analysis, where certain linguistic patterns and contextual elements can blur the line between human and machine authorship. While we acknowledge and appreciate the depth of this argument, our primary objective is to develop a model that can be directly compared to the AI detectors currently in widespread use. Given that most commercially available AI detection systems operate on a binary classification system—labeling text as

either human- or AI-generated—we find it necessary to align our approach with these existing standards.

Data:

a. Data & Algorithm
   i. The dataset we used for this project is the "Human vs. LLM Text Corpus" dataset from Kaggle. It is composed of about 788,000 samples, each with 5 features (text, source, prompt_id, text_length, and word_count). The text is the actual sample text. The source feature identifies whether the text was human written, or AI-generated (if AI-generated, the specific model is listed). The prompt_id is a primary key which is associated with the prompt used to generate the text from the source. The last two features are text length and word count of the samples. This dataset comes from Kaggle, with a usability rating of 10.00/10.00. Therefore, with the dataset being so large, and with a maximum usability rating, we are confident that it will work well for our project.
      1. https://www.kaggle.com/datasets/starblasters8/human-vs-llm-text-corpus
   ii. We had three main components as part of this project: **(1)** We created our own binary classifier. This model takes an input string, and classifies whether it is human-written or AI-generated. We used this model to compare against results that we found by testing commercial AI-detection tools. **(2)** We generated new samples that are "human-like", using a locally-hosted LLM. In order to do this, we trained and fine-tuned the parameters of a chosen model on human-written text samples, in order to generate new human-like text. Additionally, we took into account concepts like perplexity and entropy to fully understand the nuances between AI-generated and human-written text. **(3)** We then tested our human-like (yet AI-generated) text samples on the binary classifier we created.

b. Program & Platform
   i. In the implementation of the binary classification we originally debated between a logistic regression and neural network. We implemented a version using logistic regression, which is a working model with somewhat high accuracy, which will be discussed below; however, we found that we might be able to get more accuracy using a neural network. So, we ended up choosing to go with the implementation of a neural network for the problem due to the complexity and non-linear relationships that would probably be present within the dataset which a logistic regression might not be able to handle.
   ii. Based on our time and computational resources, we decided to use the TinyLlama-1.1b-chat model as it was able to generate quality text while also not being too computationally expensive to run on our computers. To make fine-tuning more efficient, we used Parameter-Efficient Fine-Tuning (PEFT) techniques,

specifically LoRA (Low-Rank Adaptation). Instead of updating all model parameters, LoRA injects small trainable matrices into specific layers, significantly reducing the number of trainable parameters. This allowed us to fine-tune the model effectively on a small dataset without, without sacrificing performance or requiring a full retraining of the base model.

c. How were experiments conducted?
    i. While training for the binary classification model, we did a train test split of 70/30 respectively, for the neural network. This meant that we had about 30% of our dataset to test against the trained model. It is also important to note that we first created a vocabulary of the most common 5000 unigram and bigram features within the dataset. This was used to train our neural network, as it provides a greater signal than plain strings of text. We also did testing for the logistic regression model to hold as a baseline for how successful our neural network is, which has an 80/20 train-test split. Results for both will be displayed in the section below.
        1. For the creation of the neural network, there were a few key steps that went into this. To begin with, all parts of the neural network were implemented from scratch, including ReLU activation function, sigmoid activation function, weight initialization, forward and backward propagation, as well as loss computations. However, after facing overfitting and unexpected behavior, we switched to PyTorch implementations to ensure we had a dependable, correct implementation. Additionally, we created a function, train_model, which was responsible for defining and training based on training data and hyperparameter arguments. We trained this neural network on the 70% training split, using Google Colab's A100 GPU. Once training was complete, the model parameters were saved and stored on GitHub. The final step of this experiment was to test the performance on the 30% testing split.
    ii. For LLM hyper-training, we used a dataset of 30 prompts designed to elicit ~1000-word responses. For each prompt, we generated 8 responses using the LLM and scored them with our Neural Network to obtain the probability that each response was human-like. The highest-scoring response from each batch of 8 was then used as the seed to prompt the LLM to generate 8 new, similar responses. This generate–score–select process was repeated for 10 rounds per prompt, creating an iterative refinement loop. We saved our Prompt-Response pairs at the end of 10 rounds and ended up with a dataset of 15 prompt-response pairs (we were not able to process more prompts because of lack of computational resources) Once we had

collected these optimized responses, we fine-tuned the TinyLlama model using this dataset to further internalize the learned patterns.

d. How to evaluate results (performance metrics)?
    i. The performance metrics that we have available for the binary classification, come in the form of using the test data set that we are working with. We will be looking to determine the accuracy of the binary classifier and logistic regression models, against commercial AI detectors. Additionally, precision, recall, and F1 score metrics will be calculated in order to determine the performance of our models. We will be producing physical graphs and tables to show the progress of our model as well as the success of our models at the states that we have them currently in the next section, results.
    ii. For the LLM, we tested its outputs by generating ~1000-word responses to a fixed set of 40 prompts both before and after fine-tuning. Each response was scored using our Neural Network to assess its probability of being classified as human. This approach allowed us to directly compare the base model's outputs to those of the fine-tuned model and evaluate the impact of fine-tuning on human-likeness.

Results:

1. Neural Network (custom binary classifier):
    a. <u>Training Parameters:</u> In order to train our custom neural network, the following parameters were used: 16 hidden dimensions, learning rate (α) of 0.01, batch size of 512 samples, and 15 epochs. These parameters were adjusted through several iterations of training, and were chosen in the end because the overall training loss converged at about 0.25, without continuing which could cause potential overfitting. Additionally a relatively standard learning rate was chosen, and a batch size small enough to ensure quick training without causing memory issues. Other important notes are: 1) random shuffling was used during training with PyTorch's DataLoader, 2) weight decay of 1e-4 was used, along with 0.3 dropout rate in order to combat overfitting.

b. Confusion Matrix: As seen in the confusion matrix below, the performance of our model based on the 30% test split resulted in 124,104 true positives, 92,066 true negatives, 12,040 false positives, and 8,467 false negatives. This can more easily be digested in the following statistics: 91.34% accuracy, 91.58% precision, 88.43% recall, and 89.98% F1 score.

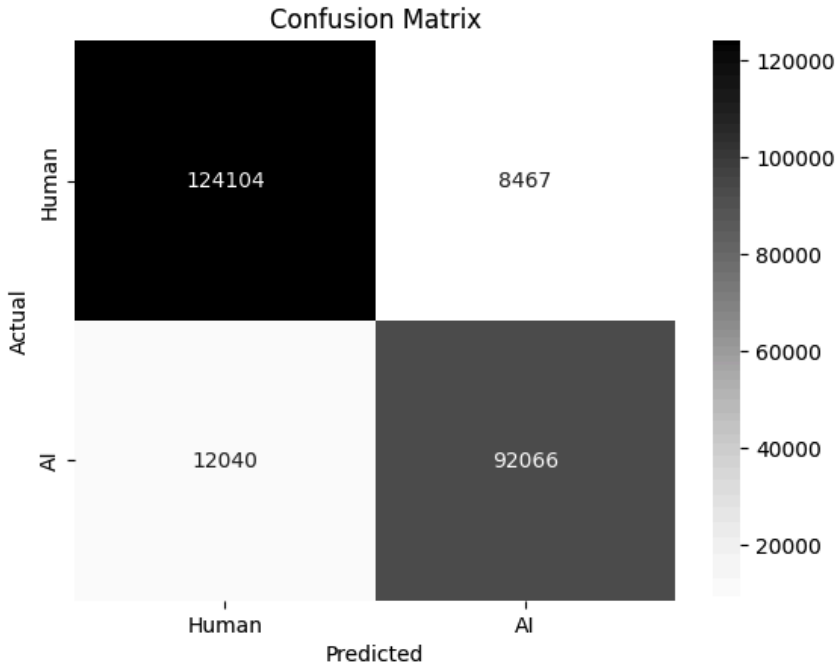


Figure 1: Confusion Matrix, derived from binary classification neural network task on 30% split of test data.

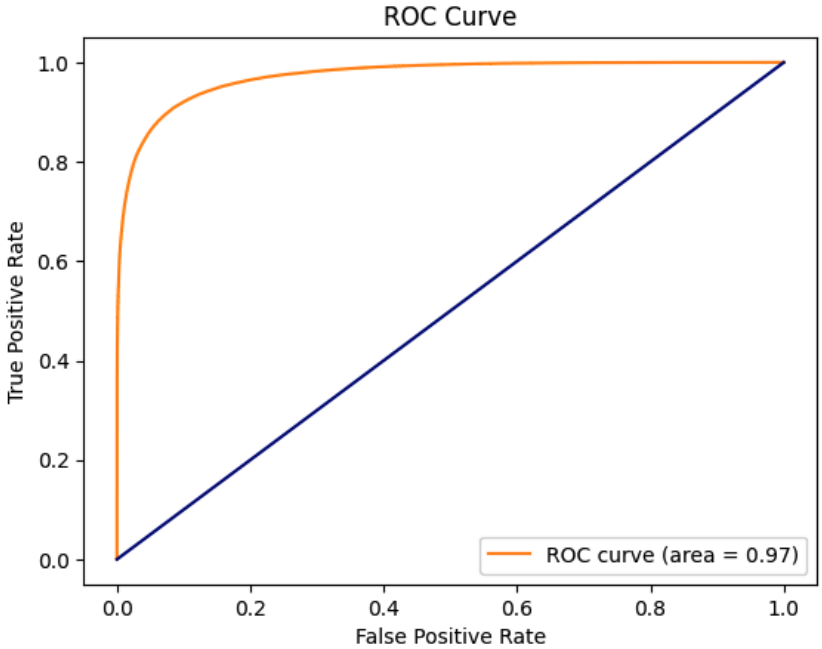


Figure 2: ROC Curve, derived from binary classification neural network task on 30% split of test data.

2. Logistic Regression:
    a. Training Parameters: In order to train our logistic regression, the following parameters were used: batch size of 64 samples (due to training constraints) and 20 epochs. These parameters were adjusted through several iterations of training, and were chosen in the end because the overall training accuracy converged at ~83%, without continuing which could cause potential overfitting. Additionally a relatively standard learning rate was chosen, and a batch size small enough to ensure quick training without causing memory issues.
    b. Confusion Matrix: As seen in the confusion matrix below, the performance of our model based on the 20% test split resulted in 79127 true positives, 51308 true negatives, 18366 false positives, and 8984 false negatives. This can more easily be digested in the following statistics: 82.67% accuracy, 85.10% precision, 73.64% recall, and 78.96% F1 score.

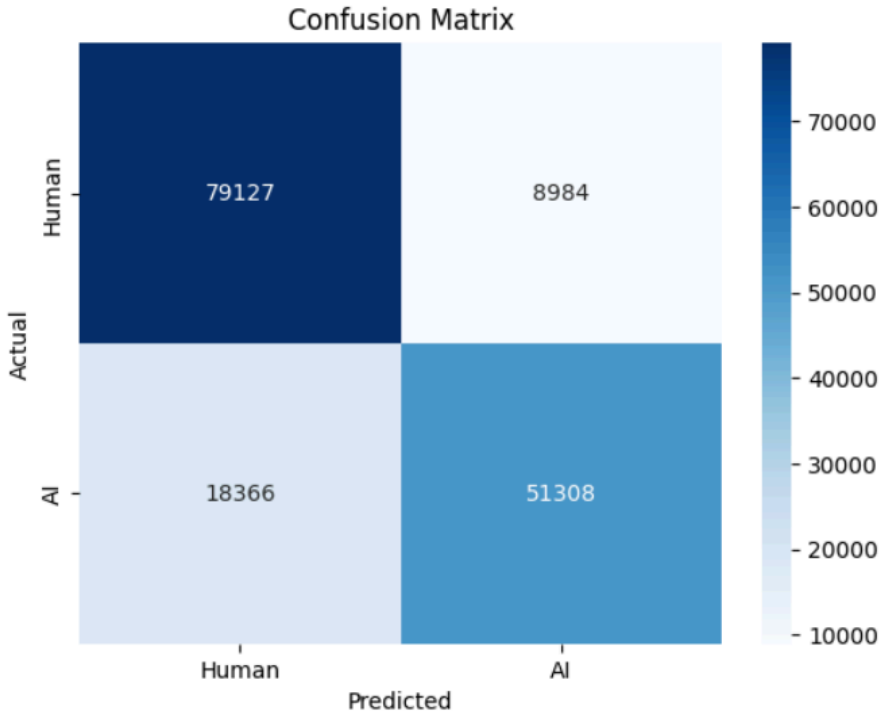


Figure 3: Confusion Matrix, derived from binary classification logistic regression task on 20% split of test data.

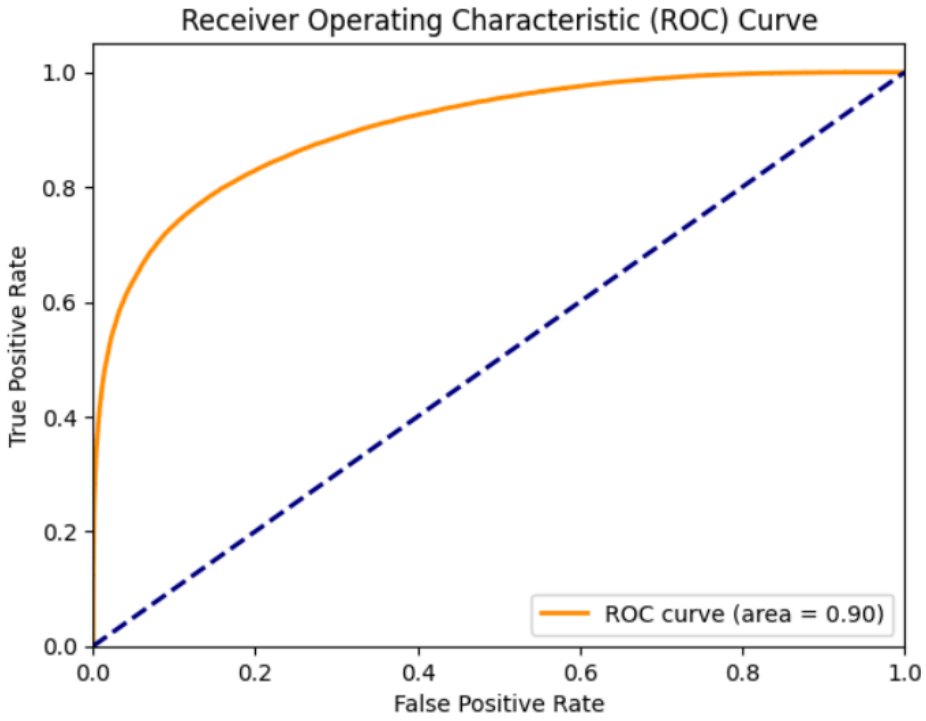
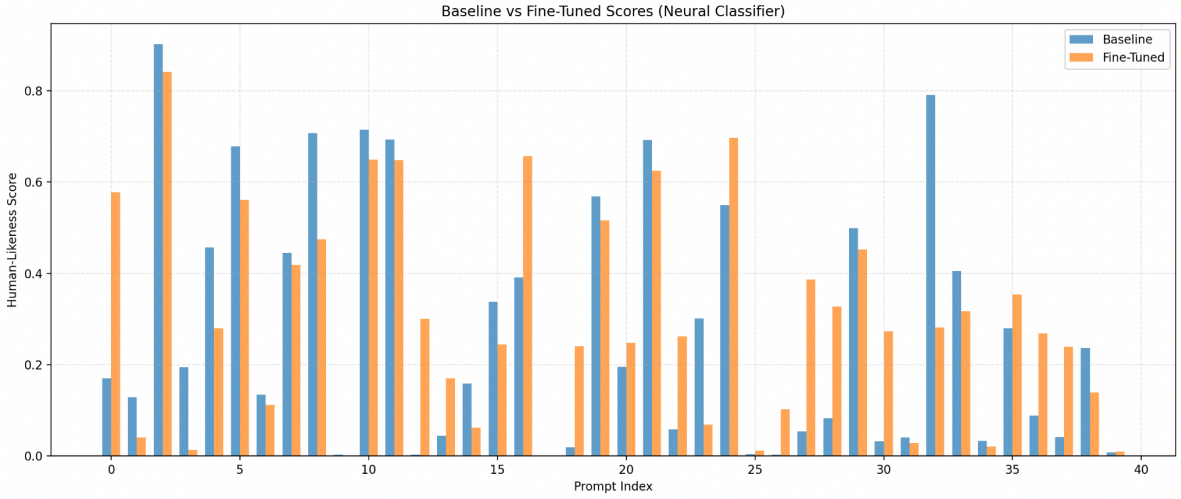


Figure 4: ROC curve, derived from binary classification logistic regression task on 20% split of test data.

3. LLM: On average, we observed a 2% increase in the probability that a response was classified as human. Notably, responses that initially scored very low showed significant improvements after fine-tuning.



Discussion:

Based on the results report in section 4 above and the objectives in section 2,

A. Neural Network (binary classifier): Overall, we were pleased with the performance of the neural network binary classifier. We were able to obtain strong performance metrics (most notably ~91% accuracy) after combating initial overfitting. The main techniques used to solve this problem were weight decay and dropout during training, as well as using a 5000 unigram and bigram vocabulary, as opposed to standard text inputs. The high accuracy of this network allowed us to confidently classify an input string as either human written or AI-generated.
B. Logistic Regression (binary classifier): Moving forward, we are still working on getting higher accuracies on the neural network, as there may not be room for improvement with the logistic regression due to training constraints. However if this model turns out to be our best bet, then we will proceed forward with this as our primary classifier.
C. LLM Fine-Tuning Outcome: While average improvement in human-likeness score was modest (~2%), our iterative generate–score–select method proved especially effective for low-scoring responses, which showed substantial gains after fine-tuning. This suggests that our approach may be most beneficial in transforming weak AI outputs into stronger, more human-like text. Based on this, we believe that by using a model with more parameters like GPT-4o (which would require massive amounts of compute resources and time) we would be able to see much better results.

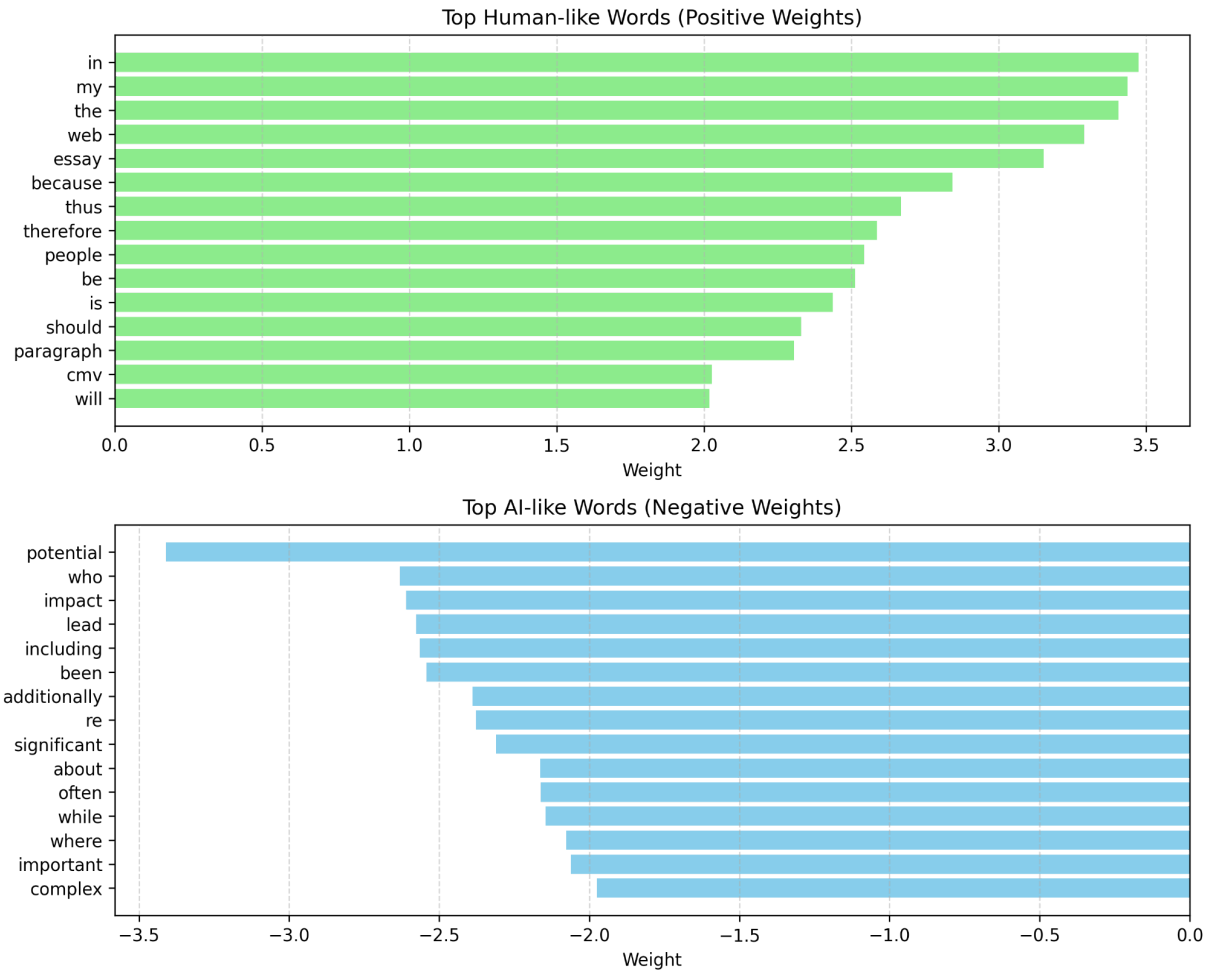


Figure 5: Distributions of the top 15 words that are human-like and ai-like based on weights.

D. **<u>For the Future:</u>** When looking to the future of this research, we think a good next step would be doing analysis on the top AI-like and top human-like words that were identified by the model. By having these words added into the training process of the LLM we would be able to enhance the human-like score of our ai-generated content. This strategy would ultimately help align us to our goal of tricking AI-detectors.

References:

a. The Effectiveness of Software Designed to Detect AI-Generated Writing: A Comparison of 16 AI Text Detectors
b. [2406.18259] Detecting Machine-Generated Texts: Not Just "AI vs Humans" and Explainability is Complicated

NOTES:

GitHub Access: https://github.com/joeskantz/ai-detection-539