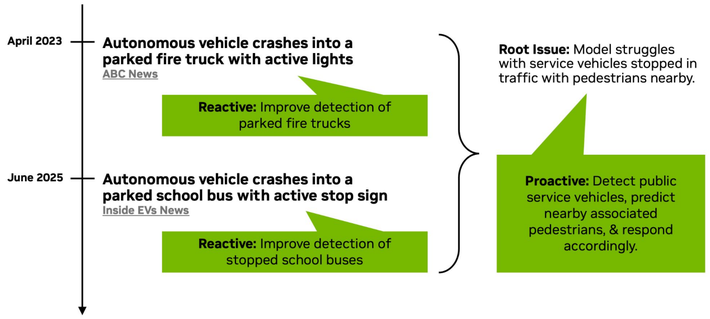
Position: Stop Reactively Patching Your Model Every Time and Start Proactive Test-Driven AI Development

Nadine Chang*, Maying Shen*, Jialiang Wang, Rafid Mahmood, Jose M. Alvarez

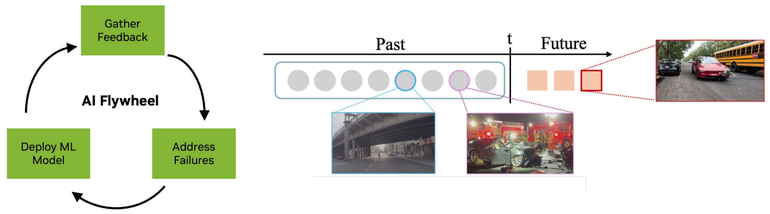
NVIDIA uOttawa

Reactive vs Proactive Coverage



Position #1: To achieve generalizable AI systems, reactive patching is structurally sub-optimal & there is a need for a goal-oriented, proactive test-driven flywheel.

Framework for Error Handling



***M** potential error scenarios* (e.g. detect fire trucks) grouped into ***K** root factors*

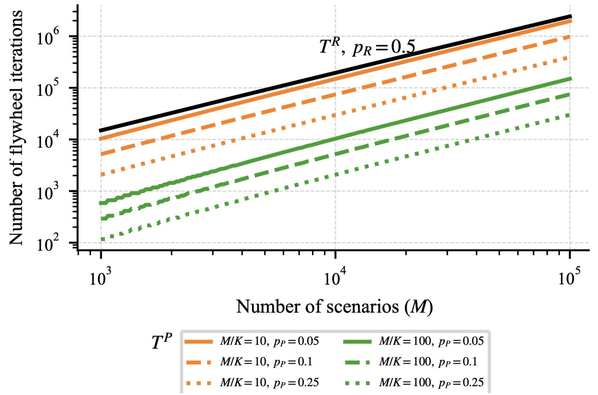
A new error is observed every iteration of an AI flywheel

Reactive policy: At each iteration, try to correct the current error with probability p_R

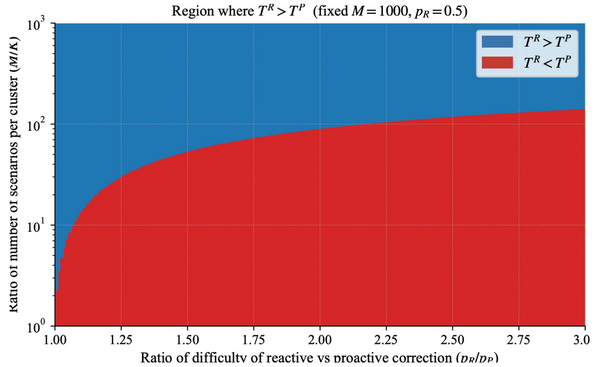
Proactive policy: At each iteration, try to correct the root factor with probability p_P

Proactive Flywheels Addresses Root Causes and Needs Fewer Iterations

We can calculate the expected time needed to correct all M errors under Reactive (T^R) and Proactive (T^P) policies.



If more scenarios share factors, then proactive correction requires orders of magnitude fewer flywheel iterations.

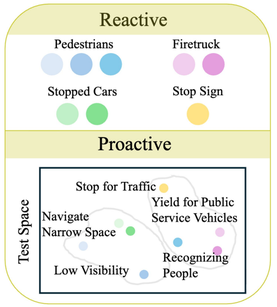


Proactive correction can be strategically optimal in the long-term even if correcting a single factor is much more difficult than correcting errors, e.g., $p_P << p_R$

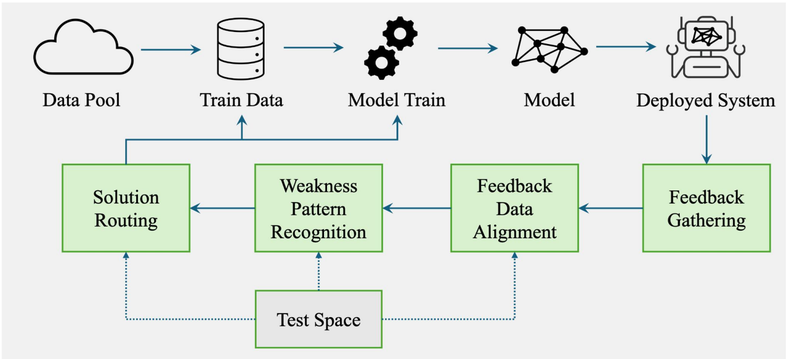
The Test Space

Test space: The space of underlying factors / test conditions governing where the system must excel

Position #2: Coverage over a test space is needed to evolve a reactive flywheel into a proactive test-driven one.



Building a Proactive Flywheel



Call for Research:

1. **How do we create a test space from known testing conditions?**
   a. What factors are implicit in the testing conditions?
   b. How can we mathematically represent root factors & their relationships?
2. **How do we align feedback using a test space?**
   a. How can we identify & prioritize factors to address?
   b. What are the encompassed factors associated with a given data sample?