On Fault Tolerance of AI Systems

Long Wang
REASONS Lab (https://reasons-lab.github.io)
Tsinghua University

The 86th IFIP WG10.4 Meeting
Gold Coast, Queensland, Australia
June 28, 2024

REASONS Lab

- From Institute for Network Sciences and Cyberspace, Tsinghua University
- Leading the REliability And Security Of Networks and Systems (REASONS) Lab
  - Focusing on reliability, security and understanding of systems, services and networks, especially when they are complicated, intelligent, autonomous, dynamic and/or software-defined
- Prior to joining Tsinghua University
  - 10+ years of research in IBM T. J. Watson Research Center on reliability and security of cloud systems
  - Department Lead of the security and reliability of IBM Watson Cloud platform
- Research interests
  - Dependable and secure systems and networks (clouds, AI systems, etc.)

REASONS Lab Members


Long Wang


Yinqin Zhao

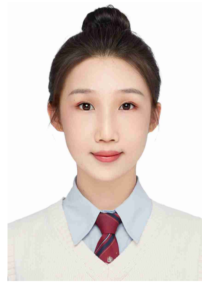

Yang Zhang


Chang Liu


Xingjian Zhang


Yiyang Chen


Mengyu Zhang


Xuanqing Shi


Yurui Gao


Mingxi Chuan


Zihou Ren


Jiayin Zhang


Haojie Wang


Gaoxu Guo


Yujiang Li

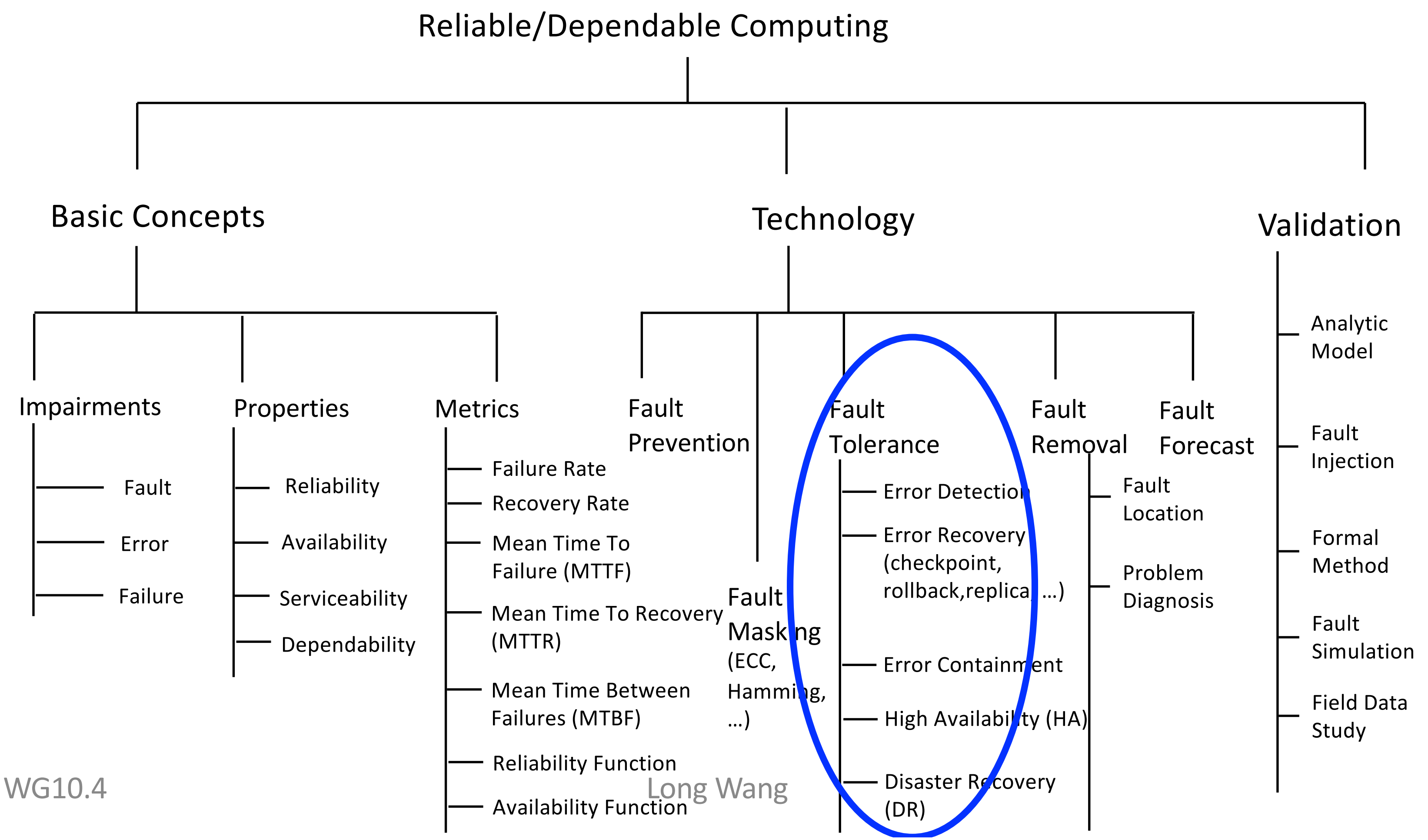

Nuoqi Gui

REliability And Security Of Networks and Systems (REASONS) Lab, Tsinghua University

Outline

- Fault Tolerance (FT) in Classical Computing
- FT of AI Systems
  - FT of AI Applications
  - FT of AI-Hosting Systems
- Case Study: FT of AIGC Applications

Overview of Classical Reliable Computing



Fault Tolerance in Classical Computing

- Fault Tolerance: The ability of a system to continue to perform its tasks after the occurrence of faults
  - Fault/Error detection
    - The process of recognizing a fault has occurred
  - Error recovery
    - The process of remaining operational or regaining operational status after the occurrence of a fault/error
  - Error containment
    - The process of isolating an error and preventing its effects from propagating throughout the system

Fault Tolerance in Classical Computing (cont.)

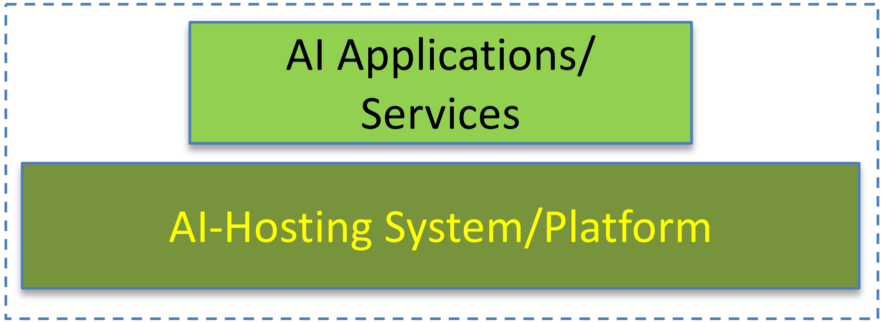
- Error Detection
  - Watchdog timers, Heartbeats
  - Consistency and capability checking
  - Exception handling
  - Control-flow checking
  - Data audits, data flow checking
- Error recovery
  - Restart
  - Checkpoint and rollback
  - Rollforward
  - Replicas/replication with failover support
- Fault tolerance
  - Hardware Redundancy
    - Triple Module Redundancy, m-out-of-n structure, active-active, active-passive
    - Voting
  - Software Fault Tolerance
    - Robust data structures
    - Recovery blocks
    - N-version programming
    - Process pair
    - Voting or Acceptance Test
  - Combining specific error detection and error recovery techniques

Outline

- Fault Tolerance (FT) in Classical Computing
- FT of AI Systems
  - FT of AI Applications
  - FT of AI-Hosting Systems
- Case Study: FT of AIGC Applications

AI Systems

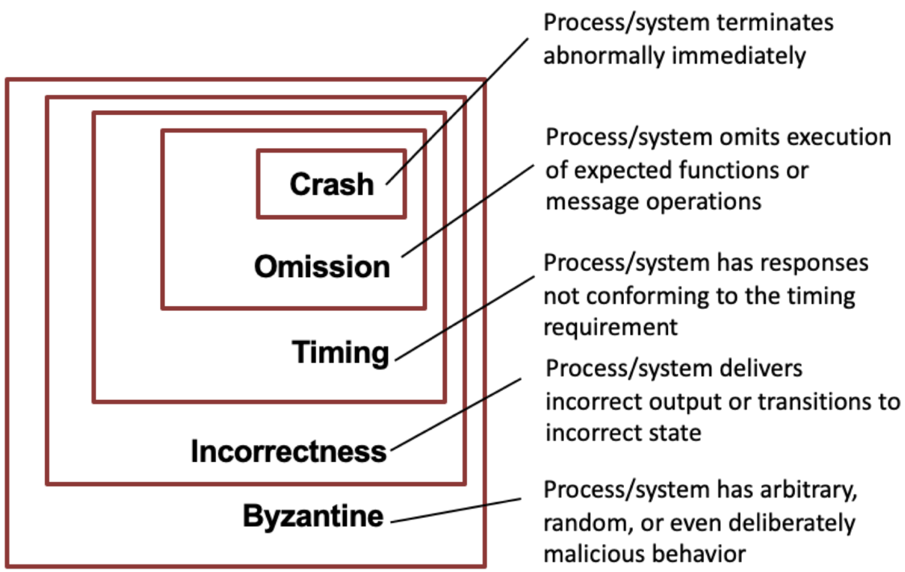
- AI systems consist of AI applications and the system/platform that hosts AI applications
- AI Applications/Services
  - Large Language Models
    - Training, content generation, inference, LLM-based fine-tuning
  - Diffusion Models
    - Training, content generation
  - Other Neural Network Models and applications
    - Classification, regression, clustering, Dimensionality Reduction
- AI-Hosting System/Platform
  - Cloud, data center, specialized big systems



AI Systems

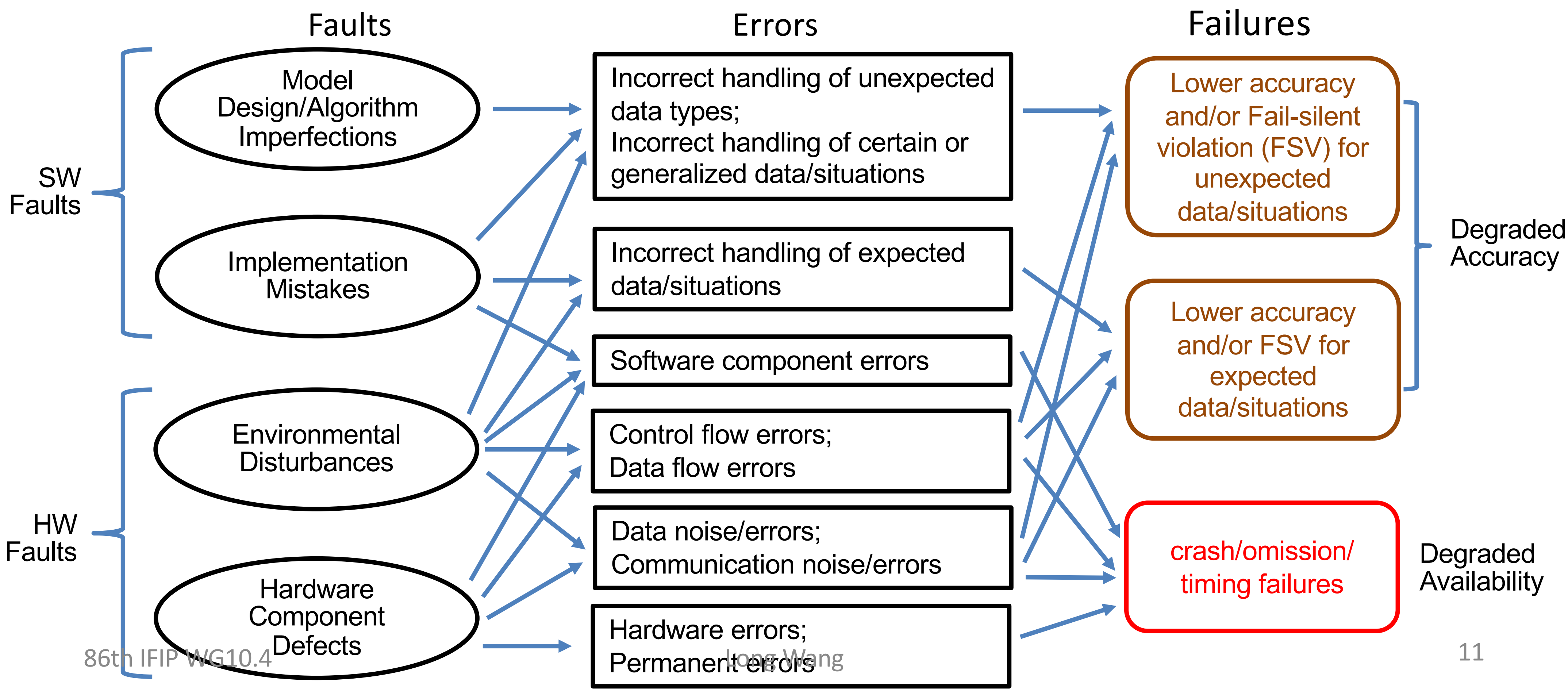
Failures of AI Systems

- Failure categories of AI systems are similar to those in classical computing
- Failures of AI applications
  - Applications do not deliver *correct output* in presence of different types of faults
  - For AI applications, *correct output* means both the **availability** and **accuracy** of the AI tasks are not degraded due to the presence of faults
- Failures of AI-Hosting systems
  - The underlying systems are subject to various failures
  - These failures incur failures of hosted AI applications
    - Incurring degraded availability of hosted AI tasks
    - Incurring degraded accuracy of hosted AI tasks, e.g. by means of communication noise



Failure Categories in Classical Computing
(for reference)

FT of AI Applications – Fault Model and Error Manifestation



FT of AI Applications – Failure Characteristics

- Degraded accuracy occurs more frequent than in classical computing
  - including fail-silence violation
- Degraded availability occurs less frequent than in classical computing
- Due to the data driven computing pattern
  - Data computation incentive
  - Has much less complex control/data flow logics
- Like classical Fault Tolerance (FT), FT of AI applications should also consider error detection and error recovery (or containment)

FT of AI Applications – Failure Model

- Degraded accuracy
  - Classification error, regression/prediction error, clustering (anomaly detection) error, generation error
  - Semantically incorrect as false positives and false negatives
  - May lead to high-impact consequences (e.g. in Autonomous Vehicles)
  - General metrics
    - precision, recall, F1 score, accuracy
  - Task-specific metrics
    - E.g. Mean Absolute Error (MAE), Mean Squared Error (MSE), or Root Mean Squared Error (RMSE) in regression tasks

- Degraded availability
  - For training
    - Long-lived
    - Single task/job
    - Akin to **failures of traditional long-lived super-computing applications**
  - For inference/generation
    - Short-lived
    - Multiple requests/jobs
    - Akin to **failures of short-lived request handlings of cloud services** that process a large number of requests

- Like classical Fault Tolerance (FT), FT of AI applications should also consider error detection and error recovery (or containment)

Fault Tolerance of AI Applications

Failure Category	Degraded Accuracy for Inference Tasks	Degraded Accuracy for Training Tasks	Degraded Availability
Error/Failure	• Incorrect task results (e.g. misclassification)	• Incorrect model states (e.g. bad model weights) • Longer convergence	• Crash, omission, timing failures • Control flow errors • Data flow errors

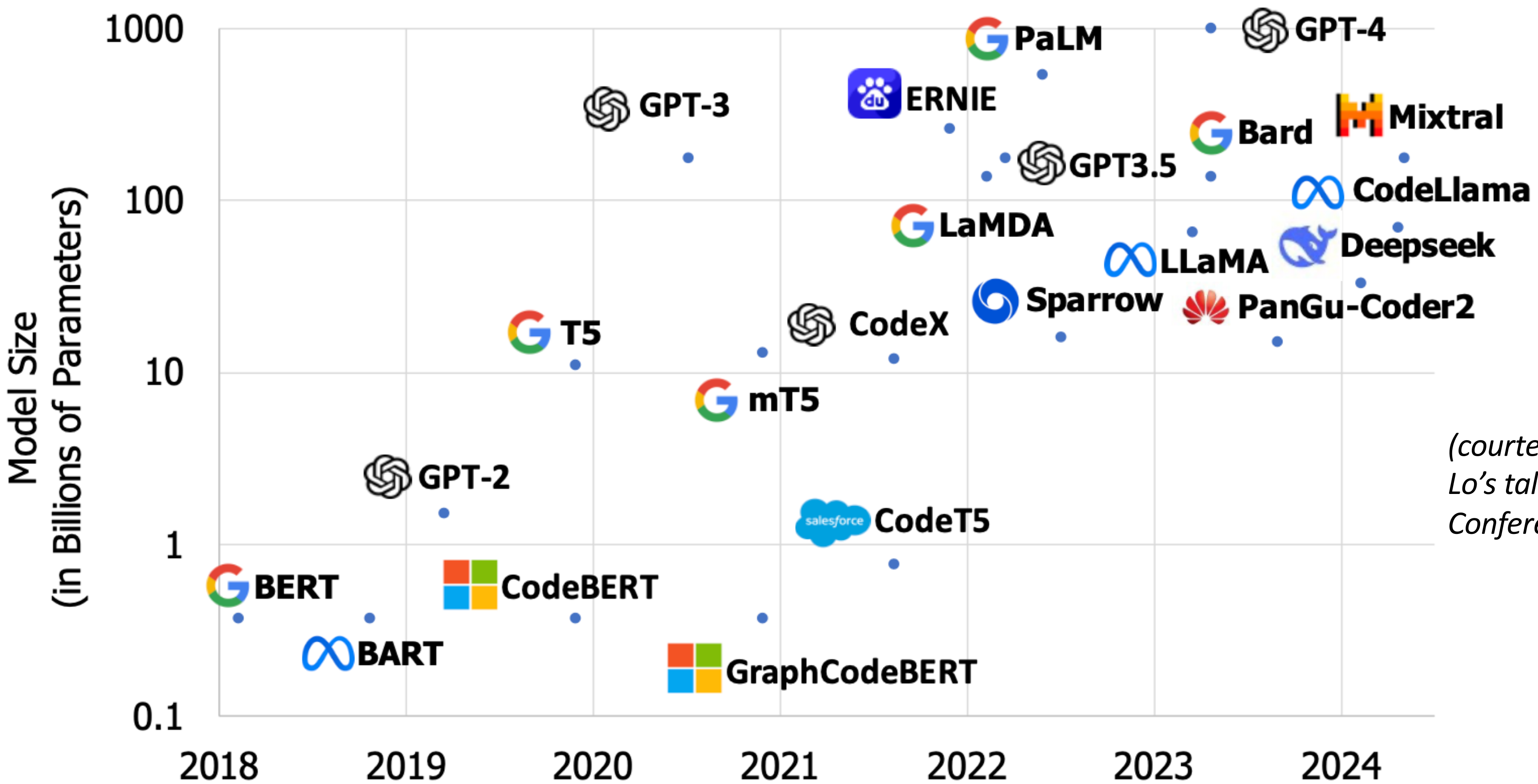
Error Detection	• The failure (incorrect result) itself
	  • E.g. user feedback
	• Acceptance check
	  • Rule based
	  • AI model based
	• Accuracy metric monitoring
		• Incorrect inference results on test data sets during training
		  • Lower accuracy
		• Range checking of model weights or intermediate values
		• Task-specific metrics for detection
			• Classical error detection
			  • Crash/hang detection, heartbeat, consistency checking, multi-replica vote, control/data flow check, data audits

Error Recovery/ Tolerance	• Re-execute, restart
	• Rollforward
	• Multi-replica vote (e.g. TMR)
	  • May be in partitioned module level
	  • Multi-version or diversified models
	• Error analysis and root-causing
	• Improving with fine-tuning
	• Re-training
		• Checkpoint/backup and rollback
		  • Saving model state periodically for recovery without losing progress
		• Multi-replica vote
		  • TMR-like
		  • Ensemble Learning
		  • May be in partitioned module level
		• Error analysis and root-causing
		• Improving with fine-tuning
			• For training tasks
			  • Checkpoint and rollback, multi-replica vote
			• For inference tasks
			  • Re-execute, restart, rollforward, multi-replica vote, replicas/replication with failover

Outline

- Fault Tolerance (FT) in Classical Computing
- FT of AI Systems
  - FT of AI Applications
  - FT of AI-Hosting Systems
- Case Study: FT of AIGC Applications

LLM Models Grow to Huge Sizes

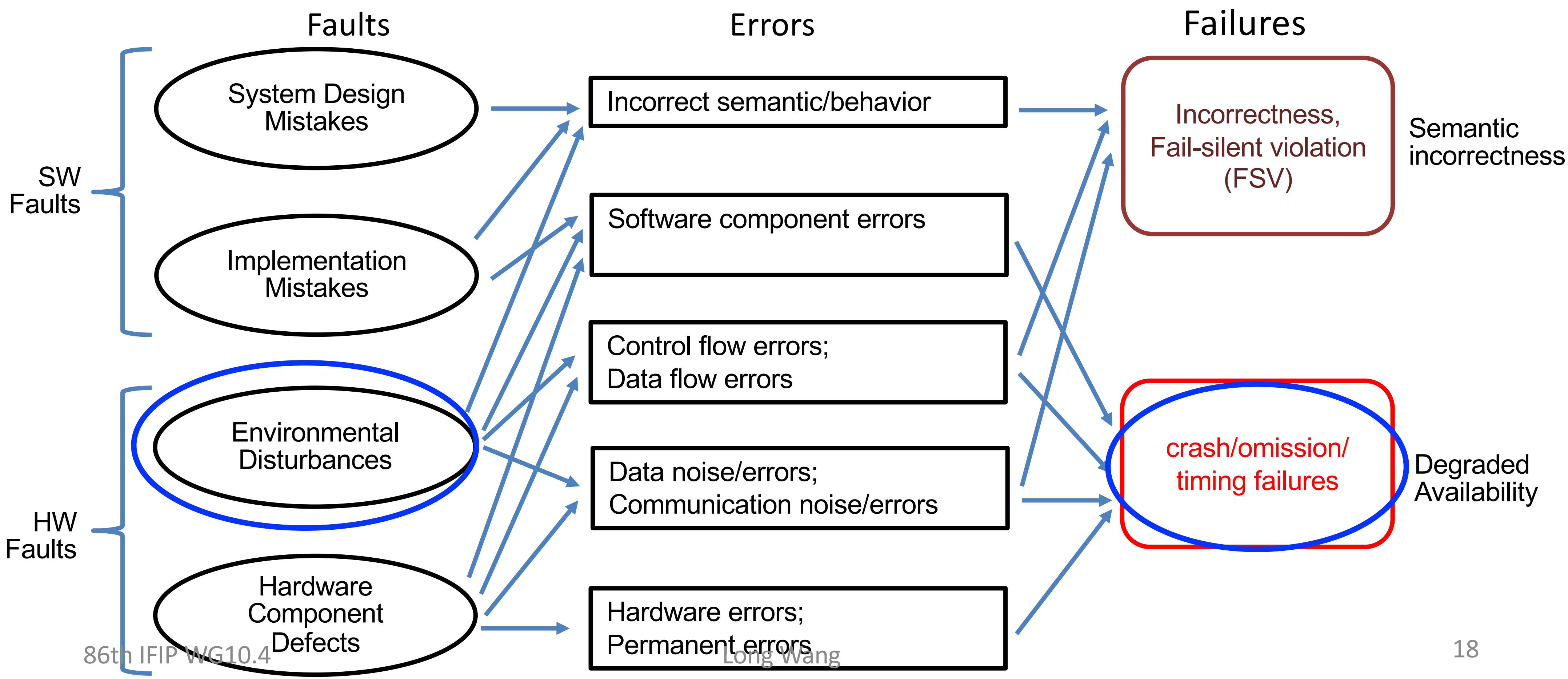


(courtesy of Prof. David Lo's talk in Huawei STW Conference 2024)

LLM-Hosting Systems Growing into Huge Sizes

- Many major cloud companies and AI companies are building AI systems with 10k~100k GPU cards
- LLM Training is a huge job using all these GPU cards
  - The training has tightly coupled logic
    - The next-step computation has dependence on the current-step computation at each GPU card
- One GPU's failure causes the entire long-term job to fail
  - Similar to traditional HPC jobs
  - Either restart the job from the beginning, or
  - Recover the job from last valid checkpoint
- High failure rates due to the huge number of GPU cards used in the job
  - Similar to the discussion in our DSN05 paper

FT of AI-Hosting Systems – Fault Model and Error Manifestation



FT of AI-Hosting Systems – Failure Model

- Failures are mostly degraded availability
  - Similar to traditional systems
  - As AI-Hosting systems are also systems, like cloud systems and data centers
- Does semantic incorrectness failure (Fail-Silent Violation) of AI-Hosting systems mostly result in degraded accuracy of hosted AI applications?
  - If so, the error detection of the degraded accuracy of hosted AI applications can be applied

Fault Tolerance of AI-Hosting Systems

- Failure Category
  - Degraded availability
  - Semantic incorrectness
- Error Detection
  - For degraded availability
    - Classical error detection (process crash, system exception, log information, error-detecting code like CRC checksum)
  - For semantic incorrectness that result in degraded accuracy, and for other semantic incorrectness
    - Error detection of AI application outputs against degraded accuracy
    - May need specific error detections (error-detecting code like CRC checksum, rule check, control/data flow check, customized check)
- Error Recovery and Tolerance
  - Similar to error recovery and tolerance of cloud systems or data centers
    - E.g. fail-forward for inference jobs, checkpoint/rollback for training jobs
  - As AI-Hosting systems are just such infrastructure as cloud systems, data centers, or simpler-structure computer systems

Fault Tolerance of AI Systems

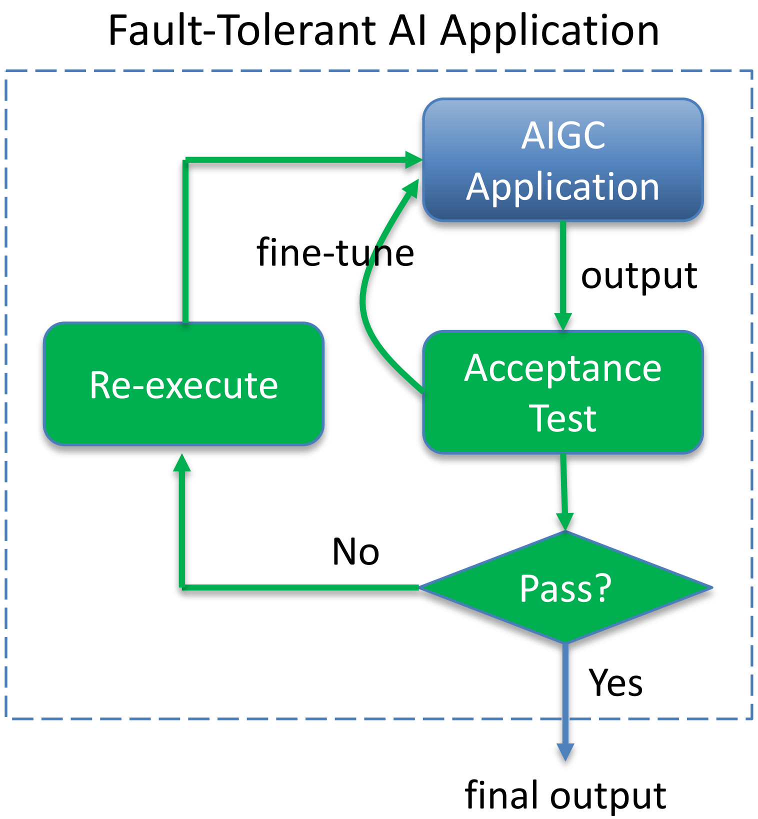
- Tolerating failures within each subsystem/component
- Detecting AI application failures and recovering the applications
  - Using proper error detection and recovery techniques
  - The recovery might be similar to that for
    - failures of traditional long-lived super-computing applications
    - failures of short-lived request handlings of cloud services
- Detecting AI-Hosting system failures and
  1. Recovering the infrastructure system first, and
  2. Shoulder-tapping the recovery of hosted applications above
- Basically, similar to fault tolerance of cloud platforms or data centers that host distributed HPC applications and cloud services

Outline

- Fault Tolerance (FT) in Classical Computing
- FT of AI Systems
  - FT of AI Applications
  - FT of AI-Hosting Systems
- Case Study: FT of AIGC Applications

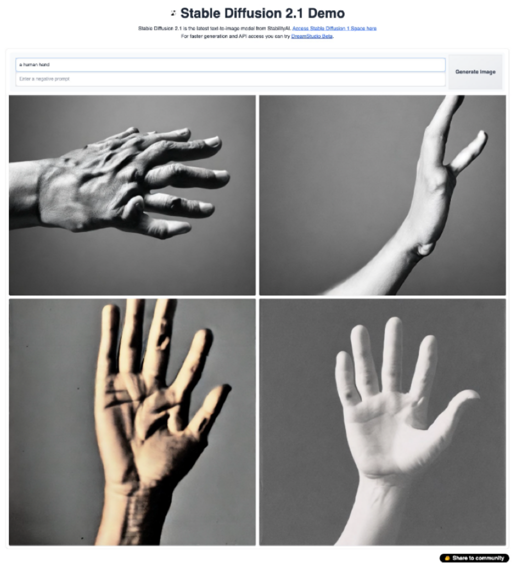
Case Study: FT of AIGC Application using Acceptance Test

- Combining error detection and error recovery for providing FT of AI applications
  - AIGC application: AI-generated content
  - Error detection: acceptance test
  - Error recovery: re-execute
- Acceptance Test
  - Rule based
    - Depending on scenarios, there may be rules that can be implemented to check if the output of is correct
  - AI model based
    - AI models as discriminators to check if the output is acceptable or not
- If the acceptance test fails, re-execute the AIGC application with different initial input
- The final output has much higher accuracy than the original one
- The acceptance test can also help fine-tune the AIGC application/model



Motivation of the Case Study

- AI generated content may have errors
- These errors may violate obvious rules, such as “a hand has five fingers”
  - A demo of Stable Diffusion 2.1 model draws “a human hand” with six fingers
- There is a need to check and regulate AI generated contents against obvious rules, as acceptance test



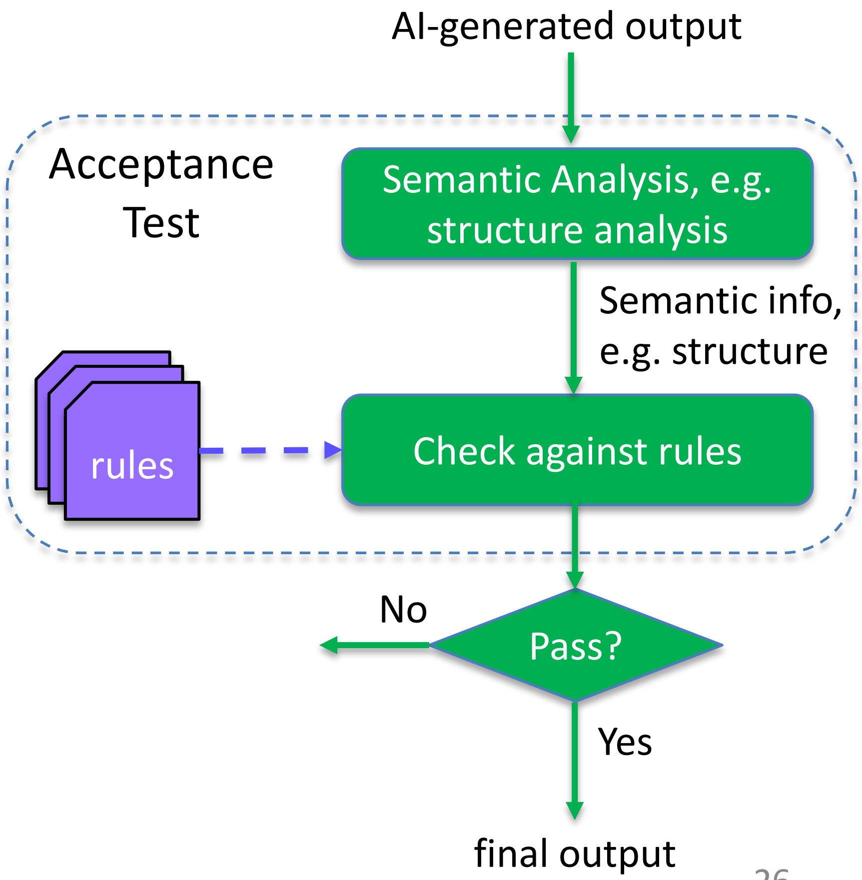
https://huggingface.co/spaces/stabilityai/stable-diffusion

Approach

- Failure model
  - Degraded accuracy (semantic incorrectness)
- Potential Faults that may incur such failures
  - Inherent imperfections of AIGC models
  - Implementation mistakes
  - Environmental disturbances
- Approach
  - We may put the rule-related information directly into the neural network training for improving the AI models
    - Like putting rule-related information into the loss function of the AI models
  - However, this way only deals with imperfections of AIGC models, and does not handle other fault types
  - A separate module of acceptance test is able to handle other two fault types that result in the failures of degraded accuracy

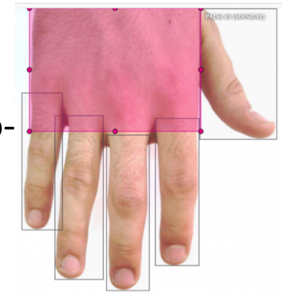
Acceptance Test Design

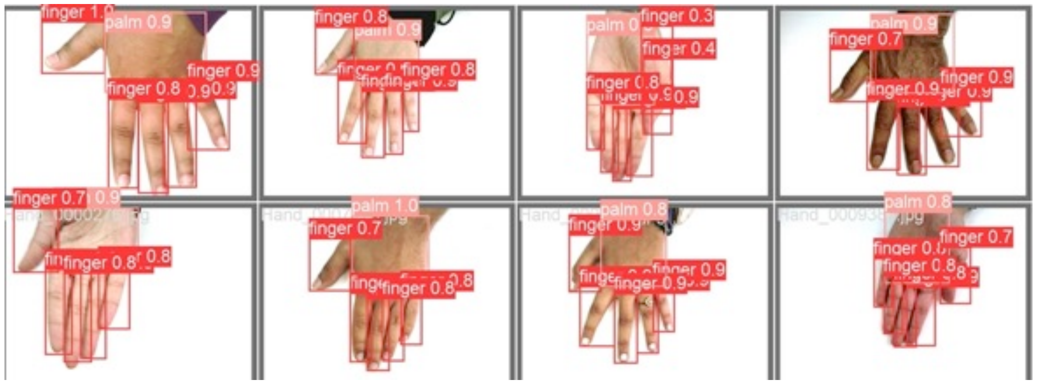
1. Semantic analysis of AI-generated output
   – e.g. analyzing the structure of the output
2. Check of the semantics against rules
   – The result of the semantic analysis is semantics information
      • e.g. the structure of content in a generated picture or a document
   – Users can specify rules based on the semantic information



An Example Acceptance Test

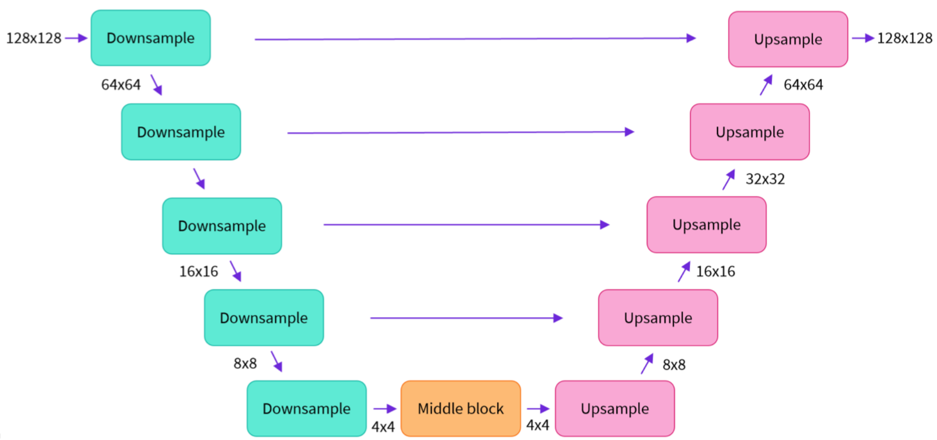
1. Structure analysis of a picture's contents
   - a hand is made up of 1 palm and 5 fingers
   - We repurpose an object-recognition and scenario-partitioning tool, YOLO (You Only Look Once), for structure analysis
     - part recognition, part partitioning
2. Check of the structure against rules
   - Semantic analysis result: the structure
     - Parts in different shapes: different types of rectangles, triangles, circles, etc.
   - "Shape of parts" rule: a hand is made up of 1 plump rectangle and 5 slim rectangles
   - The rule check: counting rectangles and checking if there is 1 plump rectangle and 5 slim ones in a hand-like object

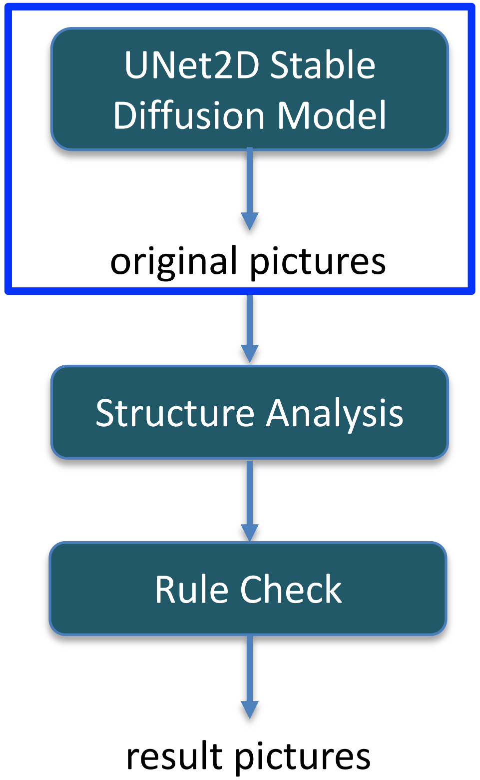




Preliminary Results (1)

- An AIGC model generates hand pictures
  - UNet2D Stable Diffusion Model
  - The generated original pictures have a high percentage of incorrect contents
    - 87.5% of generated hand pictures are correct
      - With correct number of fingers

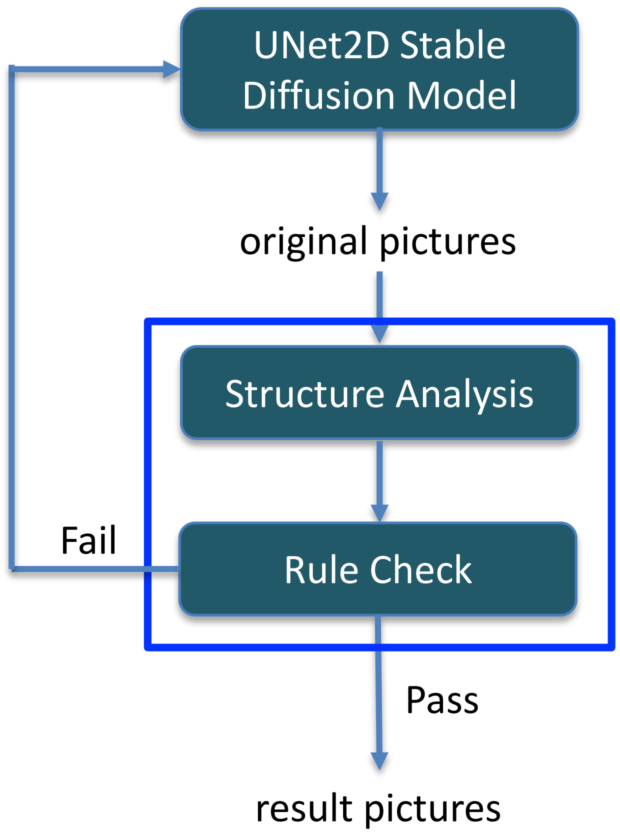




Sample data of original pictures (4 of 16 are incorrect)

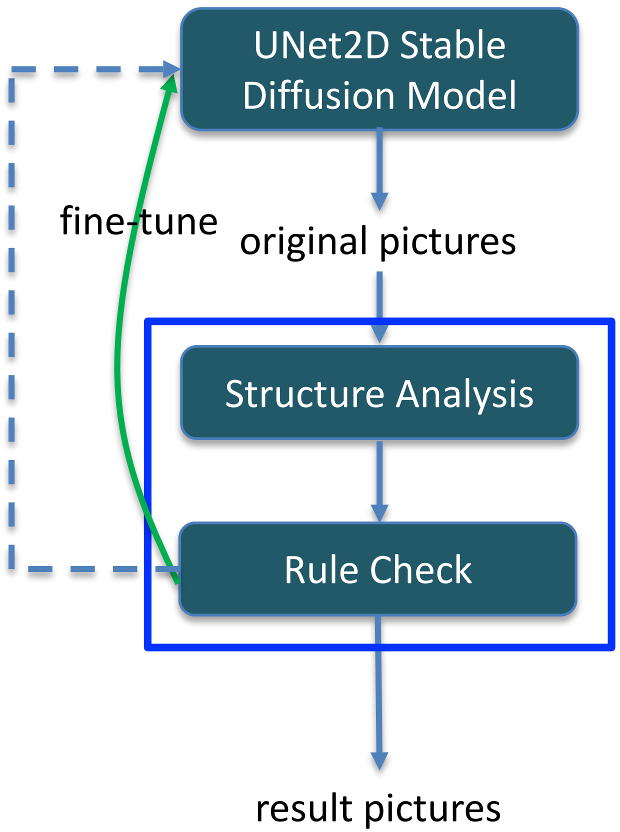
Preliminary Results (2)

- Structure analysis and rule check
  - Structure analysis: identifying object parts and their shapes
  - Rule check: counting the number of different-shape parts (5 fingers)
- The result pictures after the acceptance test-reexecution loop have 98% correctness
  - Only the pictures passing the acceptance test are delivered
  - The performance depends on the accuracy of the acceptance test (rule check)



Fine-Tuning Based on Acceptance Test

- We can fine-tune the LLM-based AIGC model
  - Leveraging the LoRA architecture
  - Using the acceptance test pass/fail output
- We can also apply the acceptance test-reexecution loop with the fine tuning



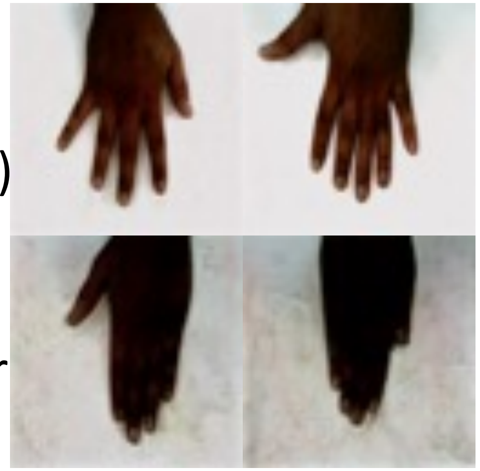
Preliminary Results (3)

AIGC Fault Tolerance Mechanism	Percentage of correct pictures (correct finger numbers)	Percentage of clear pictures
Original Stable-Diffusion (Baseline)	87.5%	88%
Fine Tuning Using Acceptance Test	92.5%	90%
Acceptance Test-Reexecution Loop	98%	99%
Fine Tuning + Acceptance Test-Reexecution Loop	99%	99%

correct incorrect

clear (sharp)

unclear



Preliminary Results (4)

- The performance of model fine-tuning is much worse than that enforced by classical acceptance test
- Moreover, environmental disturbances cannot be dealt with by model improvement (fine-tuning)
  - Model improvements only deal with the fault types of System Design Mistakes and Implementation Mistakes
- AI community always emphasize on the model improvement, but that is not sufficient for tolerating errors in AI applications/systems

Summary

- FT technologies in classical computing mostly still applies to AI applications/ systems (with adaptations if needed)
  - Error detection, error recovery, and a combination of them
  - E.g. acceptance test largely improves the AI application accuracy
- Failure models of AI applications/systems mainly fall into two categories
  - Degraded accuracy and degraded availability
- Semantic analysis based rule checking helps detect degraded accuracy of AI applications
- We can learn a lot from experiences of FT in cloud and supercomputer systems for FT of AI applications/systems, because
  - AI applications share a lot of similarities with supercomputing applications or cloud services
  - AI-hosting systems share a lot of similarities with cloud systems and supercomputer systems