N-VERSION PROGRAMMING FOR ML COMPONENTS

ARPAN GUJARATI (MPI-SWS, GERMANY)
SATHISH GOPALAKRISHNAN, KARTHIK PATTABIRAMAN (UBC, CANADA)
WHAT IS N-VERSION PROGRAMMING (NVP)?

- Software engineering principle to improve the reliability of software operations by building in fault tolerance through redundancy

**SPECIFICATION**
- Functions
- Data types

**N PROGRAMS INDEPENDENTLY DEVELOPED**
- Engineering teams that do not interact
- Different algorithms & programming languages

**N-VERSION EXECUTION ENVIRONMENT (NVX)**
- Making the N versions look like one whole system
- Redundancy suppression, e.g., using voting
WHAT IS N-VERSION PROGRAMMING (NVP)?

Software engineering principle to improve the reliability of software operations by building in fault tolerance through redundancy

- Specification
- N programs independently developed
- N-version execution environment (NVX)

- Functions
- Data types
- Engineering teams that do not interact
- Different algorithms & programming languages
- Making the N versions look like one whole system
- Redundancy suppression, e.g., using voting

Likelihood of common specification errors or common errors of interpretation

N-version programs are not statistically independent, e.g., they may fail dependently on “difficult” inputs

Given the limitations, the time and cost of developing the N versions as well as deploying an NVX is not worth!

Beautiful but fallacious theory!
Observation

\[ \text{NVP for programmed components} \neq \text{NVP for ML components} \]
 Unlike programmed components, **ML components are trained**
  - i.e., using supervised, unsupervised, or reinforcement learning

 Generating diverse ML components doesn’t require extra programming effort, but only extra computations
  - ML frameworks such as PyTorch, TensorFlow, and TVM can generate ML models with different execution plans
  - DNNs can be trained with different network structures (e.g., image recognition using ResNet50 and ResNet101)
  - Ensemble techniques can be used to train models with distinct random choices
NEW OPPORTUNITIES

► Generate and execute **hundreds of diverse replicas** inside an NVX

► **Improve the baseline reliability** of ML components, which is relatively low
  ► For example, reliability of programmed components is typically measured in “nines”
  ► In contrast, an inference accuracy of 75% – 90% is common among DNNs

Need to investigate the problem and the benefits of **NVP for ML components** with a fresh perspective!
NEW OPPORTUNITIES

- Generate and execute **hundreds of diverse replicas** inside an NVX

- **Improve the baseline reliability** of ML components, which is relatively low
  - For example, reliability of programmed components is typically measured in “nines”
  - In contrast, an inference accuracy of 75% – 90% is common among DNNs

Need to investigate the problem and the benefits of **NVP for ML components** with a **fresh perspective**!

THIS WORK

- **Mathematical modeling** to illustrate the benefits of NVP for ML components
KEY CONTRIBUTIONS
KEY CONTRIBUTIONS

▸ **Reliability modeling** in the presence of **permanent faults**, capturing
  ▸ ML components with baseline reliability *under 100%*
  ▸ NVX with *hundreds* of versions or ML component replicas
  ▸ *Parameterized diversity* percentage among each pair of replicas
  ▸ *Sequential* and *concurrent* execution semantics
  ▸ Redundancy suppression using *voting quorums* of different sizes
KEY CONTRIBUTIONS

▸ **Reliability modeling** in the presence of **permanent faults**, capturing
  ▸ ML components with baseline reliability **under 100%**
  ▸ NVX with *hundreds* of versions or ML component replicas
  ▸ *Parameterized diversity* percentage among each pair of replicas
  ▸ *Sequential and concurrent* execution semantics
  ▸ Redundancy suppression using *voting quorums* of different sizes

▸ **Numerical evaluation** using MNIST digit classification and TIMIT speech recognition tasks

NVP with tens to hundreds of replicas can significantly improve the baseline reliability of ML components

Reliability gains are sensitive to the NVX design and the diversity percentage
1. APPROXIMATION USING EXPONENTIAL FUNCTIONS

- Baseline reliability of an ML component in the presence of $x$ permanent faults:

$$R(x) = \alpha e^{-\beta x} \ (\alpha < 1)$$
1. APPROXIMATION USING EXPONENTIAL FUNCTIONS

- Baseline reliability of an ML component in the presence of $x$ permanent faults:

$$R(x) = \alpha e^{-\beta x} \quad (\alpha < 1)$$

Fault-free reliability $R(0)$ less than 100%
2. IDENTITY & DIVERSITY SUBCOMPONENTS

- In practice, without any replication, i.e., with $N = 1$
2. IDENTITY & DIVERSITY SUBCOMPONENTS

- In practice, without any replication, i.e., with $N = 1$

- We logically decompose each ML component into two parts

![Diagram showing two subcomponents: ML_identity and ML_diversity, connected with a weighted geometric mean to produce a classification output.](image-url)
2. IDENTITY & DIVERSITY SUBCOMPONENTS

- In practice, without any replication, i.e., with \( N = 1 \)

- We logically decompose each ML component into two parts

  - Remains same across replicas
  - Common cause faults cause correlated failures

- Weighted Geometric Mean

\[ \text{Weighted Geometric Mean} \]
In practice, without any replication, i.e., with $N = 1$.

We logically decompose each ML component into two parts.

- Remains same across replicas
  - Common cause faults cause correlated failures

- Varies across replicas
  - Common cause faults cause independent failures

Weighted Geometric Mean

2. IDENTITY & DIVERSITY SUBCOMPONENTS
2. IDENTITY & DIVERSITY SUBCOMPONENTS

- In practice, without any replication, i.e., with $N = 1$

- We logically decompose each ML component into two parts

- Remains same across replicas
  - $\Rightarrow$ Common cause faults cause correlated failures

- Varies across replicas
  - $\Rightarrow$ Common cause faults cause independent failures

- Weights capture the diversity percentage among each pair of replicas

- Input $\rightarrow$ ML COMPONENT $\rightarrow$ Classification Output

- $\text{ML identity subcomponent}$

- $\text{ML diversity subcomponent}$

- Weighted Geometric Mean
2. IDENTITY & DIVERSITY SUBCOMPONENTS

- In practice, without any replication, i.e., with $N = 1$

- We logically decompose each ML component into two parts

- Remains same across replicas
  - Common cause faults cause correlated failures

- Varies across replicas
  - Common cause faults cause independent failures

In short, parameterized & quantifiable diversity!

- Weights capture the diversity percentage among each pair of replicas
EVALUATION
**EXPERIMENT METHODOLOGY**

Reliability decays exponentially with faults

\[ R(x) = \alpha e^{-\beta x} \]

Baseline ML component reliability in the presence of \( x \) faults


Curve fitting using non-linear least squares
EXPERIMENT METHODOLOGY


Reliability decays exponentially with faults

Curve fitting using non-linear least squares

$R(x) = \alpha e^{-\beta x}$ Baseline ML component reliability in the presence of $x$ faults

Denoting the baseline reliability of each subcomponent using $R(x)$

$\forall n \in \{1, 2, \ldots, N\} : R_{n,\text{identity}}(x) = R_{n,\text{diversity}}(x) = R(x)$
**EXPERIMENT METHODOLOGY**


Reliability decays exponentially with faults

- $R(x) = \alpha e^{-\beta x}$ Baseline ML component reliability in the presence of $x$ faults
- $\forall n \in \{1, 2, \ldots, N\} : R_{n,\text{identity}}(x) = R_{n,\text{diversity}}(x) = R(x)$

Using combinatorial analysis

- Sequential vs. Concurrent execution — whether permanent faults cause correlated failures in the identity subcomponents
- Quorum size of $\min(2, N)$ vs. a majority quorum size of $\lfloor N/2 + 1 \rfloor$

Denoting the baseline reliability of each subcomponent using $R(x)$

Curve fitting using non-linear least squares

Evaluate the composite NVX reliability for different configurations
RESULTS USING TIMIT (quorum size of \( \min(2, N) \), diversity percentage 50%)

\[ R(x) = 77.4 e^{-0.11x} \]
RESULTS USING TIMIT (quorum size of \( \min(2, N) \), diversity percentage 50%)

\[ R(x) = 77.4 e^{-0.11x} \]

\[ R_{NVX, seq}(x), N \in [2, 32] \]
RESULTS USING TIMIT (quorum size of $\min(2, N)$, diversity percentage 50%)

$$R(x) = 77.4 e^{-0.11x}$$

- $R_{NVX, seq}(x), N \in [2, 32]$
- $R_{NVX, seq}(x), N \in [33, 64]$
RESULTS USING TIMIT (quorum size of $\min(2, N)$, diversity percentage 50%)

\[ R(x) = 77.4 e^{-0.11x} \]

- $R_{NVX,seq}(x), \ N \in [2, 32]$
- $R_{NVX,seq}(x), \ N \in [33, 64]$
- $R_{NVX,con}(x), \ N \in [2, 32]$
RESULTS USING TIMIT  
(quorum size of $\min(2, N)$, diversity percentage 50%)

\[ R(x) = 77.4 e^{-0.11x} \]

- - -  
\[ R_{NVX, seq}(x), N \in [2, 32] \]

\[ R_{NVX, seq}(x), N \in [33, 64] \]

\[ R_{NVX, con}(x), N \in [2, 32] \]

\[ R_{NVX, con}(x), N \in [33, 64] \]
RESULTS USING TIMIT (quorum size of $\min(2, N)$, diversity percentage 50%)

Increasing $N$ always helps!

$R(x) = 77.4 e^{-0.11x}$

- $R_{NVX, seq}(x), N \in [2, 32]$
- $R_{NVX, seq}(x), N \in [33, 64]$
- $R_{NVX, con}(x), N \in [2, 32]$
- $R_{NVX, con}(x), N \in [33, 64]$
RESULTS USING TIMIT (quorum size of min(2, N), diversity percentage 50%)

Increasing N always helps!
Concurrent NVX is more reliable (no correlated failures)

- - - \( R(x) = 77.4 e^{-0.11x} \)

\( R_{NVX, seq}(x), N \in [2, 32] \)

\( R_{NVX, seq}(x), N \in [33, 64] \)

\( R_{NVX, con}(x), N \in [2, 32] \)

\( R_{NVX, con}(x), N \in [33, 64] \)
RESULTS USING TIMIT (quorum size of min(2, N), diversity percentage 50%)

\[ R(x) = 77.4 e^{-0.11x} \]

- \( R_{NVX, seq}(x), N \in [2, 32] \)
- \( R_{NVX, seq}(x), N \in [33, 64] \)
- \( R_{NVX, con}(x), N \in [2, 32] \)
- \( R_{NVX, con}(x), N \in [33, 64] \)

After certain number of faults, sequential NVX start outperforming some concurrent NVX configs.
Results using TIMIT (quorum size of \(\min(2, N)\), diversity percentage 50\%)

Concurrent NVX can help significantly boost the baseline reliability

\[
R(x) = 77.4 e^{-0.11x}
\]

- \(R_{NVX,seq}(x), N \in [2, 32]\)
- \(R_{NVX,seq}(x), N \in [33, 64]\)
- \(R_{NVX,con}(x), N \in [2, 32]\)
- \(R_{NVX,con}(x), N \in [33, 64]\)
RESULTS USING TIMIT
(different quorum sizes and diversity percentages)
(quorum size of \( \min(2, N) \), diversity percentage 50%)
RESULTS USING TIMIT

(different quorum sizes and diversity percentages)

(quorum size of \( \min(2, N) \), diversity percentage 50%)

1. Quorum size of \([N/2 + 1]\) (simple majority)
RESULTS USING TIMIT

(different quorum sizes and diversity percentages)

(quorum size of \( \min(2, N) \), diversity percentage 50%)

1. Quorum size of \( \lfloor N/2 + 1 \rfloor \) (simple majority)

With a larger quorum size, more replication helps only up to a certain number of faults.
RESULTS USING TIMIT (different quorum sizes and diversity percentages)
(quorum size of \( \min(2, N) \), diversity percentage 50%)

1. Quorum size of \( \lceil N/2 + 1 \rceil \) (simple majority)

With a larger quorum size, more replication helps only up to a certain number of faults

2. Varying the diversity percentage (\( N = 32 \))
RESULTS USING TIMIT

(different quorum sizes and diversity percentages)
(quorum size of \( \min(2, N) \), diversity percentage 50%)

1. Quorum size of \( \lceil N/2 + 1 \rceil \) (simple majority)

With a larger quorum size, more replication helps only up to a certain number of faults.

2. Varying the diversity percentage (N = 32)

By introducing sufficient diversity, even sequential NVX can offer higher reliability.
SUMMARY

- Historically, NVP has faced criticism!
- NVP for ML components is different, needs to be revisited
  - There is potential to significantly improve ML component reliability
  - Our mathematical modeling demonstrated some of these benefits
- Future work
  - Does our logical decomposition hold in practice? Test using simulations, FI
  - Can we achieve such high replica diversity? Is the diversity quantifiable?
  - NVX design space (including voting schemes) need to be explored further
SUMMARY

- Historically, NVP has faced criticism!
- NVP for ML components is different, needs to be revisited
  - There is potential to significantly improve ML component reliability
  - Our mathematical modeling demonstrated some of these benefits
- Future work
  - Does our logical decomposition hold in practice? Test using simulations, FI
  - Can we achieve such high replica diversity? Is the diversity quantifiable?
  - NVX design space (including voting schemes) need to be explored further

THANK YOU! QUESTIONS?