Machine Learning Verification: Student Notes

By Brian Hyeongseok Kim

CSCI 698, Fall 2025

Slide 1 — Machine Learning Verification

This course introduces the emerging field of *Machine Learning (ML) Verification*. As ML models are increasingly deployed in sensitive, safety-critical, and socially impactful settings, we must ask: **Can we guarantee their correctness?**
Verification aims to provide mathematically sound assurances of model behavior, going beyond what testing alone can offer.

Slide 1 — Software is Everywhere

Modern life is built on software. From computers and communication to aviation, software systems operate behind the scenes, including safety-critical systems.

Slide 2 — Software Testing

Traditionally, software reliability is evaluated through software testing, where you check whether a property ϕ holds for some given input.

The slide illustrates a faulty implementation of a max(a, b) function, where both branches assign result = a, yet several test cases *still pass*. This highlights a critical limitation: testing is incomplete.

Slide 3 — Software Verification

Verification asks a stronger question: Does ϕ hold for all possible inputs? That is, can we guarantee their correctness?

Rather than checking only sampled cases, verification attempts to reason over the entire input space. In the max(a,b) example, verifying an invariant such as, assert(result ≥ a && result ≥ b) quickly exposes the bug.

While this assertion is not enough, verification always guarantees a defined behavior. It is up to the developer to define what behavior they want to verify.

Slide 4 — ML is Everywhere

Similarly, machine learning systems are now embedded broadly in society. Because ML models make decisions on data, they influence real people and real systems, which raises fairness, safety, and robustness concerns. So, applying similar pipeline of testing and verification becomes important.

Slide 5 — ML Workflow and Testing

Typical ML pipeline:

1. Data — raw information collected from the world
2. Training — the model learns patterns from the data
3. Model — the trained network
4. Inference — applying the model to new inputs to produce outputs

Testing an ML model means checking whether ϕ holds for some sampled input(s). Just like traditional software testing, this approach may miss important bugs or undesirable behaviors because ML input spaces are enormous and sampling is limited. Testing alone cannot guarantee fairness or safety.

Slide 6 — ML Verification

Verification in ML asks the stronger question: Does ϕ hold for all inputs? Similarly, we must consider verification not only as a software issue but also as a model behavior issue.

But in ML, “input space” can involve extremely high-dimensional continuous vectors. *Verification aims to certify properties such as fairness and robustness across every possible input in a defined domain. This is much harder than verifying classical software,* yet increasingly essential.

Slide 7 — Possible Definitions of ϕ

The slides list four important properties often studied in ML verification:

1. Counterfactual Fairness
   Prevents differences in output when only protected attributes change.
2. Epsilon Fairness
   A fairness notion defined over small neighborhoods in input space.
3. Local Robustness
   Ensures that small perturbations around a specific input do not change its label.
4. Global Robustness
   Ensures robustness for *all* inputs over the entire domain.

Each defines ϕ differently, capturing fairness or robustness from different perspectives.

Slide 8 — Counterfactual Fairness

Counterfactual fairness concerns *protected attributes* (e.g., gender, race).

A model is counterfactually fair if:

For any input x, modifying only its protected attribute (to obtain x') does not change the output.

This is verified over *the entire domain*.

Referenced work: **Kim et al., ICSE 2025**.

Slide 9 — Epsilon Fairness

Epsilon fairness is limited to *local neighborhoods*.

For a given x, any x' within ε distance and with the *same protected attribute* should not have a different label.

This property focuses on fairness among “similar individuals” by some distance metric.

Referenced work: **Li et al., CAV 2023**.

Slide 10 — Local Robustness

Local robustness is a standard ML verification property:

Within an ε-ball around an input x, the model output should not change.

This is commonly used in adversarial robustness research to ensure that small perturbations do not lead to misclassification.

Note that this is similar to epsilon fairness, since they both pertain to local neighborhoods defined by ε distance.

Referenced work: **Leino et al., ICML 2021**.

Slide 11 — Global Robustness

Global robustness generalizes local robustness to the *entire input domain*:

No pair of points within ε distance anywhere in the domain should receive different labels.

Labels may include a special symbol ⊥ indicating "not locally robust" along the decision boundary. Two labels match under a special relation (⊥=) that treats ⊥ as acceptable when comparing.

Global robustness is very strong and difficult to guarantee, but useful for safety-critical domains.

Referenced work: **Leino et al., ICML 2021**.

Slide 12 — Conclusion

We have covered:

- The difference between *testing* and *verification*
- Why verification is essential for ML: fairness, safety, robustness
- Definitions of major verification properties (ϕ)
- Concrete examples showing when models satisfy or violate these properties

In conclusion, machine learning verification provides guarantees that testing alone cannot, which is critical as ML systems are deployed in high-stakes domains.