



Université  
de Rennes

# SEM Metamorphic Testing

FILOCHE Leo, BINDEL Jeremy, DE ZORDO Benjamin



ECOLE SUPERIEURE  
D'INGENIEURS DE RENNES

# Introduction

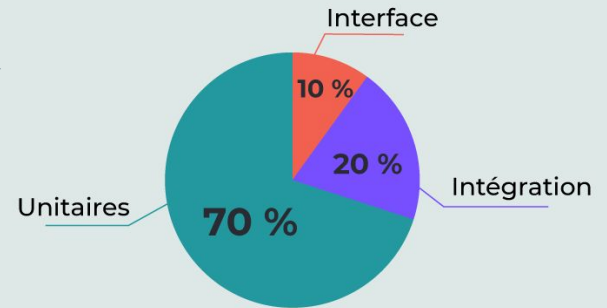
*“Pourquoi avons-nous besoin de tests dans le developpement logiciel ?”*

-> garantir la qualite et la fiabilite des systemes d'informations

Des methodes traditionnelles

*... creation d'un oracle impossible ?*

*Solution le tests metamorphiques !  
utiliser les proprietes du code a tester*





*Quelle est la portée et le champ d'application du test métamorphique  
dans l'industrie ?*



# Plan



**01**



**Introduction**

**04**



**Services web et applications**

**02**



**DeepLearning**

**05**



**Conclusion**

**03**



**Bio-informatique**

**06**



**Exemple**





**01**

**Tests métamorphiques  
dans le Deep Learning**



# Tests métamorphiques dans le Deep Learning (1/2)

## Importance du test dans le DL

Les modèles de DL actuels présentent des performances telles que de nombreuses organisations les intègrent dans leurs systèmes.

Certaines applications du DL se font dans des **systemes critiques**.

Nécessité de **garanties** pour éviter/encadrer les comportements inattendus.

Il faut détecter leurs **incohérences**, et évaluer leur **robustesse**.

## Utilisation du MT comme solution

Sur base de **relations métamorphiques**, des **transformations** sont définies.

Le modèle effectue une prédiction **avec** et **sans** la transformation

Une **comparaison** des résultats des prédictions est effectuée.

## Limites des méthodes traditionnelles de test

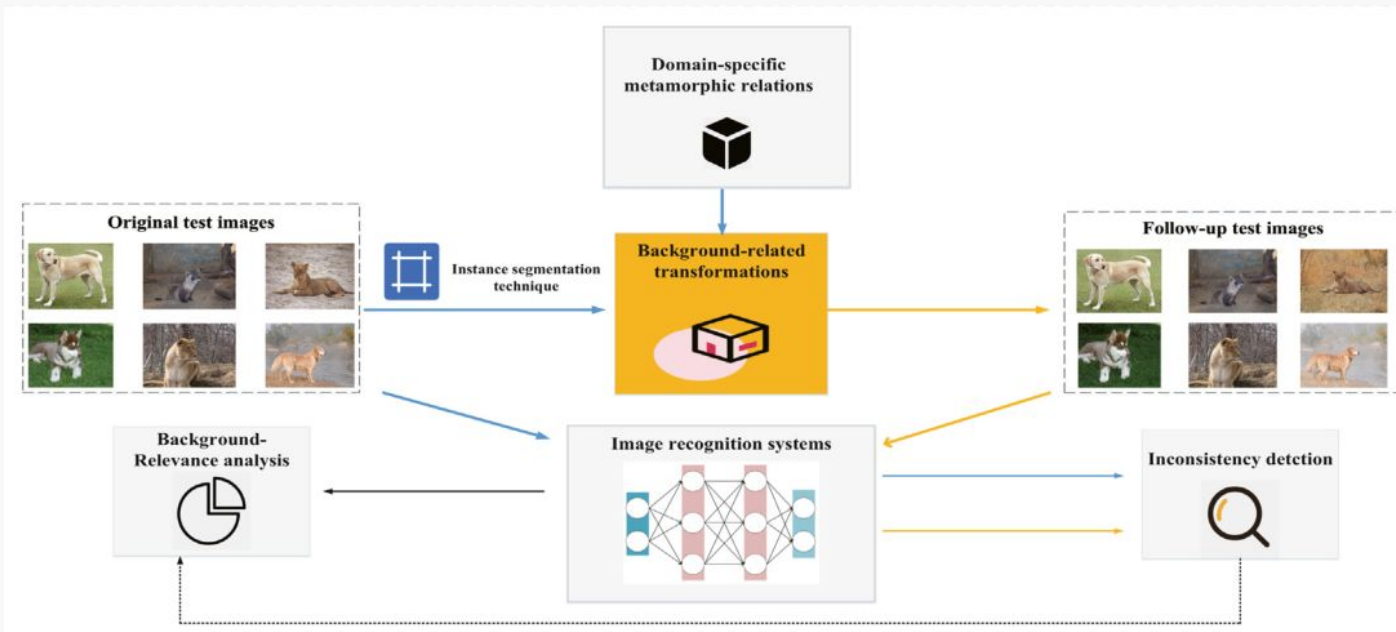
Les tests traditionnels sont basés sur les **spécifications** du logiciel.

Le DL ne propose pas de spécifications et de structure logique comme les logiciels traditionnels.

L'**espace d'entrée** et de sortie est énorme, et il est souvent difficile d'évaluer si la sortie est correcte ou non.

# Tests métamorphiques dans le Deep Learning (2/2)

## Une illustration avec DeepBackground



Objectif

Améliorer la robustesse des modèles de reconnaissance d'images.

Propriété étudiée

L'arrière plan des images.

“DeepBackground: Metamorphic testing for Deep-Learning-driven image recognition systems accompanied by Background-Relevance” - Zhiyi Zhang, Pu Wang, Hongjing Guo, Ziyuan Wang,



**02**

# **Tests métamorphiques dans la Bio-informatique**





# Tests métamorphiques dans la Bio-informatique (1/2)

---

Les logiciels de bio-informatique sont de gros artefacts :

- taille des entrees
- processus d'extraction des donnees complexes

Impossibilite de s'assurer de l'exactitude des sorties, probleme de l'oracle.

⇒ Le test metamorphique a toute sa place.

# Tests métamorphiques dans la Bio-informatique (2/2)

GNLab

Given a network G and a node P in G, we add an edge, which is directed to P with zero weight. The new network is referred to as G'. The output of G' should be identical to the output of G.

MRs	Original	GM1	GM2	GM3	GM4	GM5	GM6	GM7	GM8	GM9
MR1 (a)										
MR1 (b)										
MR2 (a)				RYE		RYE	RY	Y		
MR2 (b)				RYE	RYE		RYE			
MR2 (c)	RYE	RYE	RYE	YE	RYE	YE	RYE	RYE	RYE	RYE
MR3 (a)										
MR3 (b)										
MR4			RYE						RYE	RYE
MR5 (a)				RYE	E	E	RYE	RYE		
MR5 (b)				RYE	E	RY	RYE	RYE		

GNLab and nine of its mutants were tested against three batches of test cases, which are labeled as R (random), Y (yeast) and E (E. coli). Each pair of test cases that detects a violation of a MR in a program is labeled by its batch in the respective cell in the table. For example, the label 'RY' in the cell [GM6, MR2(a)] indicates that mutant 6 violates MR2(a) according to the test cases in batch R and batch Y.

An innovative approach for testing bioinformatics programs using metamorphic testing by Tsong Yueh Chen<sup>1</sup>, Joshua WK Ho<sup>\*2,3</sup>, Huai Liu<sup>1</sup> and Xiaoyuan Xie<sup>1</sup>



**03**

# **Tests métamorphiques sécuritaires dans les Systèmes Web**



# Tests métamorphiques sécuritaires dans les Systèmes Web (1/2)

---

Tests en securite :

- Utilise l'application entière et pas une seule fonction,
- Necessite des connaissances et un état d'esprit (que certain developpeur n'ont pas),
- Complexite des scenarios de securite (exploitation de diverses vulnerabilites pour contourner les mesures de protection).

Consequences -> oracles

# Tests métamorphiques sécuritaires dans les Systèmes Web (2/2)

Tests metamorphiques :

The image shows the Eclipse IDE interface with a Java class `OTG_AUTHZ_002` and its test results. The code defines a metamorphic test `MR()` that checks if actions are processed in a specific order. The test results in the console show a failure due to a mismatch in the order of actions.

```
OTG_AUTHZ_002.java
8
9 @SuppressWarnings("all")
10 public class OTG_AUTHZ_002 extends MR{
11     public boolean mr() {
12         List<Action> _actions = Operations.Input(1).actions();
13         for (final Action action : _actions) {
14             ifThenBlock();
15             if (!Operations.IsSupervisorOf(Operations.User(), action.getUser())) &&
16                 Operations.cannotReachThroughGUI(Operations.User(), action.getUrl()) && Operatio
17             ifThenBlock();
18             boolean_OR = Operations.OR(
19                 Operations.IsError(Operations.Output(Operations.Input(1), action.getPosition()),
20                 Operations.NOT(Operations.Output(Operations.Input(1), action.getPosition()).equals(Operations.Output(Operations.Input(2), action
21             if (_OR) {
22
```

**FAILURE:**

```
Input(2) [FOLLOW-UP INPUT] :
Input(2)=[
[Action 400661947529008 : access the index http://192.168.56.102:8080/]
[Action 400661947820045 : log in with (user!,user!Pass) at http://192.168.56.102:8080/]
[Action 400661947836931 : click on http://192.168.56.102:8080/computer/slave1/]
[Action 400661947951548 : click on http://192.168.56.102:8080/computer/slave1/launchSlaveAgent]
[Action 400661948015380 : click on http://192.168.56.102:8080/computer/slave1/]
]
Input(1) [SOURCE INPUT] :
Input(1)=[
[Action 400661947529008 : access the index http://192.168.56.102:8080/]
[Action 400661947820045 : log in with (admin,admin!Pass) at http://192.168.56.102:8080/]
[Action 400661947836931 : click on http://192.168.56.102:8080/computer/slave1/]
[Action 400661947951548 : click on http://192.168.56.102:8080/computer/slave1/launchSlaveAgent]
[Action 400661948015380 : click on http://192.168.56.102:8080/computer/slave1/]
]
User(1) [ACCOUNT INPUT] :
User(1)=(Account: username=user! pwd=user!Pass)

**Inputs processed:
Input(1) (action position: 3)
Input(1) (action position: 3)
Input(2) (action position: 3)
****

**Inputs processed:
Input(1) (action position: 3)
Input(1) (action position: 3)
Input(2) (action position: 3)
****
```

The context menu on the right shows the **JUnit Test** option highlighted, indicating the test execution process.



# Conclusion

# Conclusion sur les tests métamorphiques

Origine : 1998

Substitut au problème de l'oracle

Domaine d'application **riche et varié**



**Ne remplace pas** les autres stratégies mais fait partie du tout



# Articles

ACM Computing Surveys, “Metamorphic Testing: A Review of Challenges and Opportunities”, 2018. #Reference

ELSEVIER, “DeepBackground: Metamorphic testing for Deep-Learning-driven image recognition systems accompanied by Background-Relevance”, 2021. #Forward

BMC (Springer Nature) “An innovative approach for testing bioinformatics programs using metamorphic testing”, 2009. #Backward

IEEE, “Metamorphic Security Testing for Web Systems”, 2020. #Forward

- Github, “SMRL Project”.
- Youtube, “SMRL: A Metamorphic Security Testing Tool for Web Systems”.

Girkard AI, How to test ML Models : Metamorphic Testing.



**Exemple\* :**

**Application du MT en tant  
que développeur avec**



**Giskard**

\* : A été masqué durant la présentation

# Exemple : Utilisation du MT avec Giskard



## Giskard

Initiative Française et Open Source proposant une librairie de tests pour le ML (Python)

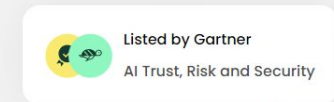
### The testing framework for ML models

Eliminate risks of biases, performance issues & security holes in ML models. In <8 lines of code.

From tabular models to LLMs

GET STARTED

TRY IT IN COLAB



```
# Get started  
pip install giskard -U
```

You can copy code here



Objectif de limiter les vulnérabilités liées au ML :

- Biais de performance
- Robustesse
- Fairness
- ...

Propose une API simple. Plusieurs méthodes de test sont disponibles, dont les tests métamorphiques

# Exemple : Utilisation du MT avec Giskard

## Exemple de test d'un modèle supervisé

```
1 from giskard import demo, Model, Dataset, testing, transformation_function
2
3 model, df = demo.titanic()
✓ 0.0s
```

PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Fare	Embarked	Survived	
123	124	2	Webber, Miss. Susan	female	32.50	0	0	13.0000	S	yes
714	715	2	Greenberg, Mr. Samuel	male	52.00	0	0	13.0000	S	no
412	413	1	Minahan, Miss. Daisy E	female	33.00	1	0	90.0000	Q	yes
81	82	3	Sheerlinck, Mr. Jan Baptist	male	29.00	0	0	9.5000	S	yes
555	556	1	Wright, Mr. George	male	62.00	0	0	26.5500	S	no
...	...	...	...	...	...	...	...	...	...	...
755	756	2	Hamalainen, Master. Viljo	male	0.67	1	1	14.5000	S	yes
442	443	3	Petterson, Mr. Johan Emil	male	25.00	1	0	7.7750	S	no
471	472	3	Cacic, Mr. Luka	male	38.00	0	0	8.6625	S	no
695	696	2	Chapman, Mr. Charles Henry	male	52.00	0	0	13.5000	S	no
664	665	3	Lindqvist, Mr. Eino William	male	20.00	1	0	7.9250	S	yes

446 rows × 10 columns

```
1 wrapped_model = Model(model=model, model_type="classification")
2
3 wrapped_dataset = Dataset(
4     df=df,
5     target="Survived",
6 )
✓ 13.7s
```

## Etape 1 : Chargement du modèle et des données de test

# Exemple : Utilisation du MT avec Giskard

## Exemple de test d'un modèle supervisé

```
1 @transformation_function(name="Increase Age")
2 def increase_age(row):
3     row["Age"] = row["Age"] * 1.2
4     return row
5
6 @transformation_function(name="Decrease Age")
7 def decrease_age(row):
8     row["Age"] = row["Age"] * 0.8
9     return row
10
11 @transformation_function(name="Reverse the sex")
12 def reverse_sex(row):
13     row["Sex"] = "male" if row["Sex"] == "female" else "female"
14     return row
```

✓ 0.0s

## Etape 2 : Ecriture de fonctions de transformation

# Exemple : Utilisation du MT avec Giskard

## Exemple de test d'un modèle supervisé

### Etape 3 : Test de l'invariance du modèle suite aux transformations

```
1 result = testing.test_metamorphic_invariance(  
2     model=wrapped_model,  
3     dataset=wrapped_dataset,  
4     transformation_function=increase_age  
5 ).execute()  
6  
7 result
```

9] ✓ 0.0s

✓ Test succeeded

Metric: 0.98

- 354 rows were perturbed

```
1 result = testing.test_metamorphic_invariance(  
2     model=wrapped_model,  
3     dataset=wrapped_dataset,  
4     transformation_function=decrease_age  
5 ).execute()  
6  
7 result
```

2] ✓ 0.0s

✓ Test succeeded

Metric: 0.99

- 354 rows were perturbed

```
1 result = testing.test_metamorphic_invariance(  
2     model=wrapped_model,  
3     dataset=wrapped_dataset,  
4     transformation_function=reverse_sex  
5 ).execute()  
6  
7 result
```

7] ✓ 0.0s

✗ Test failed

Metric: 0.22

- 446 rows were perturbed