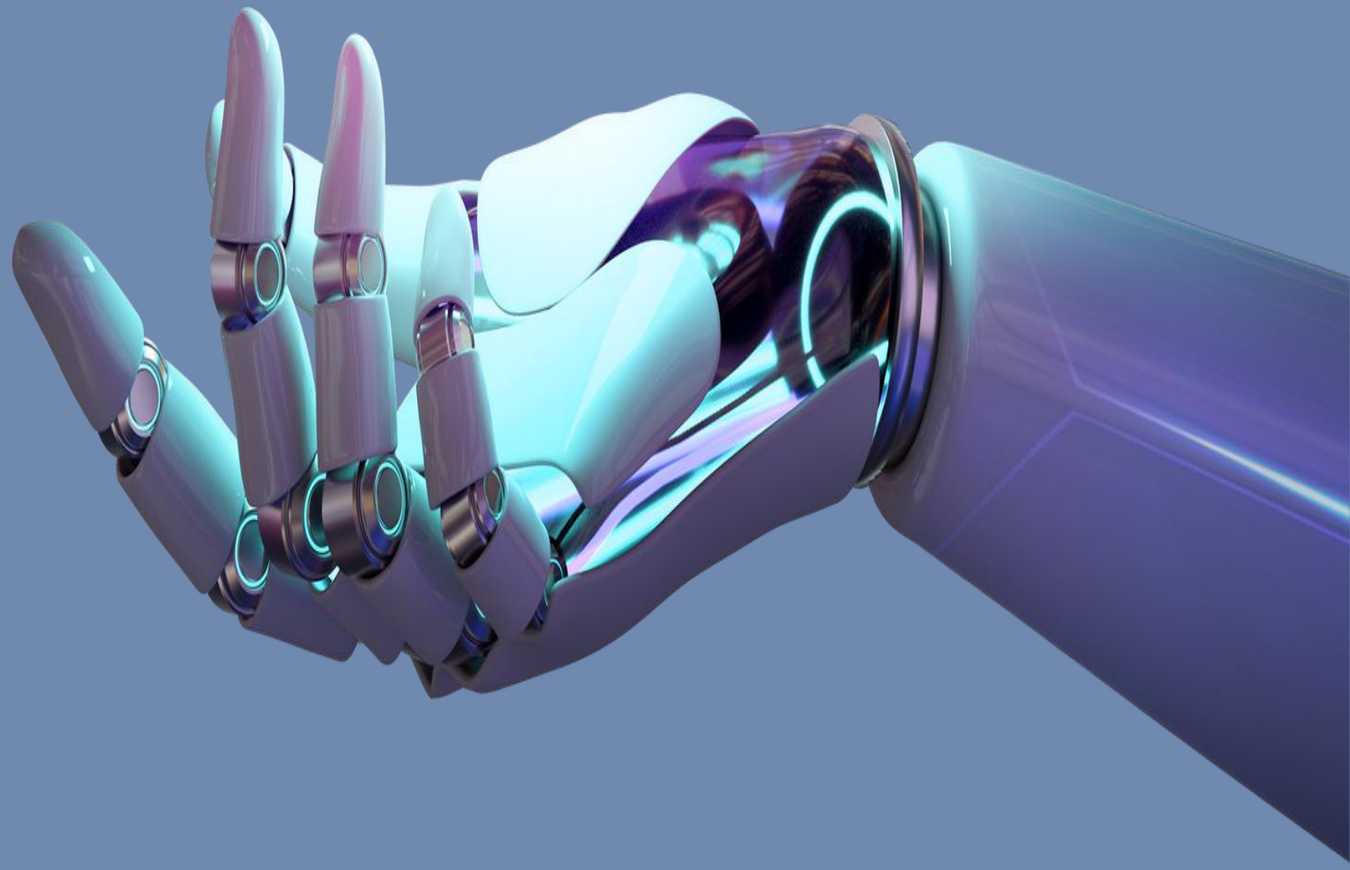
Université Sultan Moulay Slimane Faculté Polydisciplinaire Béni Mellal Département de Mathématiques et Informatique Master STRI

Programmation avec Theano

REALISER PAR:
ELARISS-ELIDRISSI MERYEM & AHESSAB LHOSSAINE
ENCADRANT :
PROF ANOUAR DARIF

P L A N :

- C'est quoi Theano ?
- Pourquoi Theano ?
- Les domaines d'utilisation ?
- Comment installer Theano ?
- Les fonctions de Theano
- Des exemples
- Conclusion



C’est quoi Theano :

Theano est une librairie Python qui permet de définir, d'optimiser et d'évaluer des expressions mathématiques, notamment celles comportant des tableaux multidimensionnels. Theano est un compilateur d'algèbre linéaire .

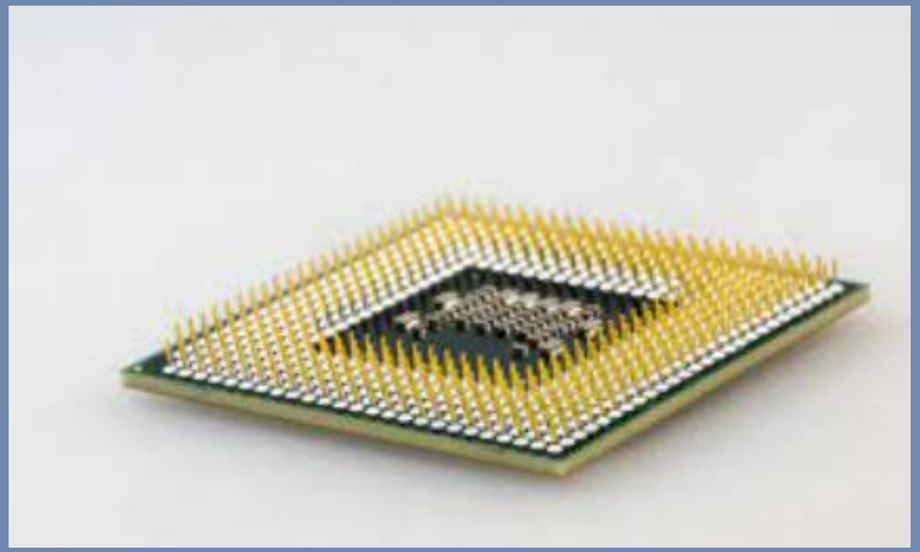
il est nommé d’après un mathématicien grec (Theano (philosophe)). Theano a été développé à l’Université de Montréal , dans un groupe dirigé par Yoshua Bengio, depuis 2007.

Theano peut réduire au minimum les frais généraux de compilation/analyse, mais fournit toujours des fonctionnalités symboliques telles que la différenciation automatique.

À l'aide de Theano, il est possible d'atteindre des vitesses comparables à des implémentations C artisanales pour des problèmes impliquant de grandes quantités de données. Il peut aussi dépasser C sur un processeur GPU de plusieurs ordres de

GPU

&



CPU

Pourquoi Theano :

La bibliotheque Theano a de nombreuses caracteristique :

Intégration étroite avec NumPy– Utilisez numpy.ndarray dans les fonctions compilées Theano.

Utilisation transparente d’un GPU – Effectuer des calculs à forte intensité de données beaucoup plus rapidement que sur un CPU.

Optimisation de la vitesse et de la stabilité – Obtenez la bonne réponse pour log(1+x), même lorsque x est vraiment minuscule.

Différenciation symbolique efficace – Theano réalise vos dérivés pour des fonctions avec une ou plusieurs entrées.

génération de code C dynamique – Évaluez les expressions plus rapidement.

tests unitaires approfondis et auto-vérification – Détecter et diagnostiquer de nombreux types d’erreurs....

Les domaine d’utilisation de theano:

Theano est une bibliothèque utilisée dans plusieurs domaines parmi ceux-ci trouvent

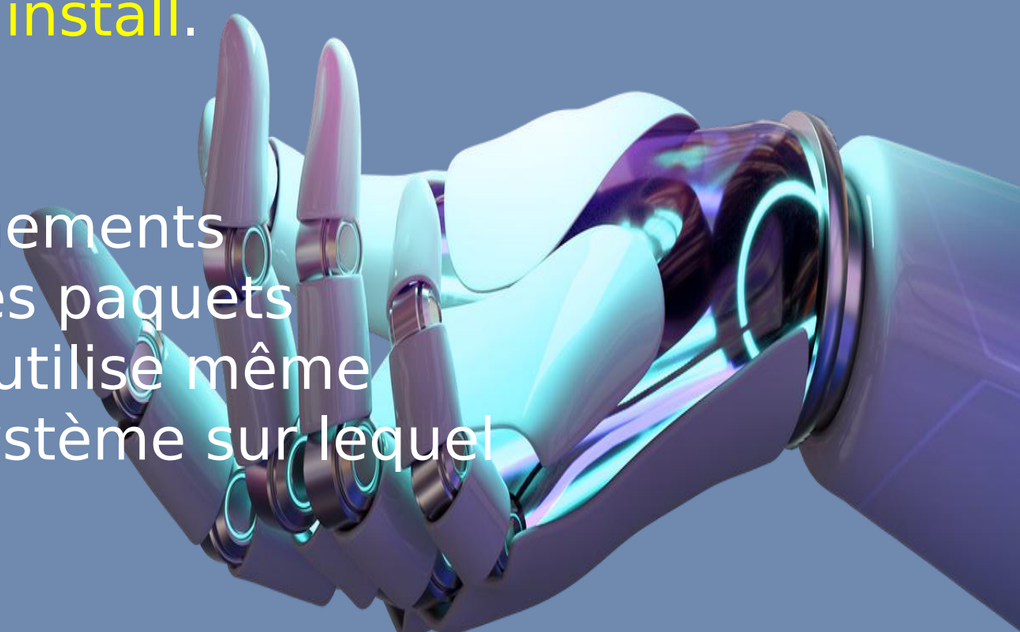
la recherche scientifique, l'intelligence artificielle, l'apprentissage profond (Deep learning)...

Comment installer Theano :

La façon la plus simple d'installer Theano est d'utiliser conda, un paquet multi-plateforme et 'environment manager'.
Theano est maintenant disponible sur PyPI (*Python Package Index*) et peut être installé via easy_install Theano, pip install Theano ou en téléchargeant et déballant l'archive et en tapant python setup.py install.

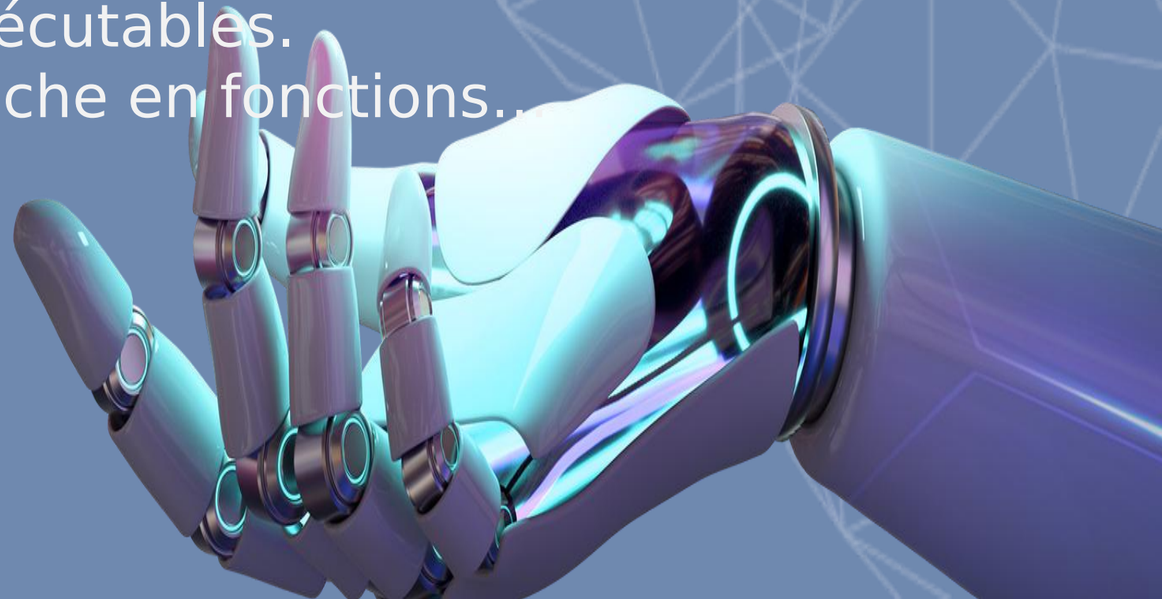
Remarque important :

Conda nous permet de créer de nouveaux environnements dans lesquels des versions de Python (2 ou 3) et les paquets installés peuvent différer. L'environnement conda utilise même version de Python que la version installée sur le système sur lequel vous conda installé.



Les fonctions en Theano

Ce module "Theano" fournit ses fonctions couramment utilisées sous la forme de theano.function(), l'interface permettant de compiler les graphiques en objets exécutables.
Theano, Il s'agit d'une bibliothèque riche en fonctions...



Les tenseurs en Theano :

La force de Theano consiste à exprimer des calculs symboliques faisant intervenir des tenseurs (tensor). Il existe de nombreux types d'expressions symboliques pour les tenseurs. Elles sont regroupées dans les sections suivantes :

- 0-Order tensor

Scalars

- 1-Order tensor

Vectors

- 2-Order tensor

Matrices

- Multi-dimensional array 3D

Tensors

Syntax : mavariable = theano.tensor.tensor_type('mavariable')

Les TYPEs des TENSEUR :

Il existe de nombreux types de tenseurs, donnons quelques exemples importants :

Constructeur	dtype	ndim
bscalar	int8	0
wscalar	int16	0
iscalar	int32	0
lscalar	int64	0
fscalar	float32	0
dscalar	float64	0
cscalar	complex64	0
zscalar	complex128	0
fvector	float32	1
ivector	int32	1
fmatrix	float32	2
ftensor3	float32	3
dtensor3	float64	3

Prefix	dtype
b	int8
w	int16
i	int32
l	int64
f	float32
d	float64
c	complex64
z	complex128

Les operateurs sur les TENSEUR :

On peut appliquer les opérateurs sur n'importe quel type de tenseur comme suite. :

Avec T = theano.tensor :

Opérateur	Description
T.add, T.sub, T.mul, T.truediv	Adition, subtract, multiplication, division
T.pow, T.sqrt	La puissance, racine_caree
T.exp, T.sin,T.tanh,T.cosh…	Les fonctions trigonometriques
T.intdiv, T.mod	La division int, modulos
T.sgn	sign
T.and_ , T.xor , T.or_ ….	& , ^ ,| ….
T.gt , T.lt , T.ge , T.le	Comparaison
T.isnan	Comparaison avec NaN(not a number)
T.abs_	La valeur absolut
T.minimum ,T.maximum	Maximum et minimum
T.clip	couper les valeurs entre un maximum et un minimum

Expression scalar:

```
1  import theano
2  import theano.tensor as T
3  a = T.dscalar('a')
4  f = theano.function([a],a**2)
5  x = eval(input('entrez un nombre : '))
6  print( f(x) )
```

Output

```
In [1]: runfile('C:/Users/HASSNA/.spyder-py3/temp.py', wdir='C:/Users/HASSNA/.spyder-py3
WARNING (theano.tensor.blas): Using NumPy C-API based implementation for BLAS functions.

entrez un nombre : 7
49.0

In [2]:
```

Nous pouvons retourner plusieurs valeurs d'une fonction :

```
1  import theano
2  import theano.tensor as T
3  a = T.dscalar('a')
4  f = theano.function([a],[a**2,a**5])
5  x = eval(input('entrez un nombre : '))
6  print( f(x) )
```

Output

```
In [1]: runfile('C:/Users/HASSNA/.spyder-py3/temp.py', wdir='C:/Users/HASSNA/.spyder-py3
WARNING (theano.tensor.blas): Using NumPy C-API based implementation for BLAS functions.

entrez un nombre : 4
[array(16.), array(1024.)]

In [2]:
```

La sortie est un tableau « array » avec le carré et la puissance 5 du nombre

Vectors et Matrix Expression :

```
1  import theano
2  import theano.tensor as T
3  a = T.vector()
4  b = T.vector()
5  out = a**2 + 2*a*b + b**2
6  f = theano.function([a, b], out)
7  print(f([3,2], [4,2]))
8
```

Output

```
In [1]: runfile('C:/Us
WARNING (theano.tensor
[49. 16.]

In [2]:
```

```
1  #exemple d'une logistique fonction
2  import theano
3  import theano.tensor as T
4  x = T.dmatrix('x')
5  s = 1/(1+T.exp(-x))
6  logistic = theano.function([x],s)
7  print(logistic([[0,1],[2,4]]))
8
```

Output

```
In [4]: runfile('C:/Users/
[[0.5        0.73105858]
 [0.88079708 0.98201379]]

In [5]:
```

La fonction SCAN( ) :

Les fonctions scan fournissent les fonctionnalités de base nécessaires pour réaliser des boucles dans Theano. Scan est livré avec de nombreux sifflets et cloches, que nous allons introduire à titre d'exemples.

```python
import theano
from theano import tensor as t,scan,function
le=10
h1=0
h2=1
seq=[]
def step(h1,h2):
    h=h1+h2
    return h,h1
for i in range(le):
    h1,h2=step(h1,h2)
    seq.append(h1)
print("---------- python pure ----------")
print(seq)
print("---------- avec theano ----------")
hh1,hh2=t.scalars('h1','h2')
outputs, updates = theano.scan(step,sequences=[],outputs_info=[hh1,hh2],n_steps=le)
F=function([hh2,hh1],outputs)
print(F(0,1))
```

Output

```
In [32]: runfile('C:/Users/hajar/Desktop/exercice pytho/exemple/untitled4.py',
wdir='C:/Users/hajar/Desktop/exercice pytho/exemple')
---------- python pure ----------
[1, 1, 2, 3, 5, 8, 13, 21, 34, 55]
---------- avec theano ----------
[array([ 1.,  2.,  3.,  5.,  8., 13., 21., 34., 55., 89.]), array([ 1.,  1.,  2.,
3.,  5.,  8., 13., 21., 34., 55.])]
```

La fonction Shared :

La fonction shared construit des variables dites partagées. Il s’agit de variables hybrides symboliques et non symétriques dont la valeur peut être partagée entre plusieurs fonctions.

```
1   import theano
2   import numpy
3
4   x = theano.tensor.dmatrix('x')
5   W = theano.shared(numpy.asarray([[0.2, 0.7,4],[2,5,0]]), 'W')
6   y = (x * W).sum()
7
8   f = theano.function([x], y)
9
10  output = f([[0.2, 0.7,3],[1,5,0]])
11  print (output)
12
```

Output

```
In [17]: runfile('
39.53

In [18]:
```

Gradient en Theano :

La fonction de gradient est utilisée pour calculer la dérivée d'une expression mathématique,

Le calcul de gradient est l'un des aspects les plus importants de la formation d'un modèle d'apprentissage profond.

Syntaxe simple : output = tensortyp.grad(expression,variable)

Exemple 1 :

```
1   import theano
2   import theano.tensor as T
3   x = T.dscalar('x')
4   y = x**3
5   output = T.grad(y,x)
6   f = theano.function([x],output)
7   print(f(4))
8   # cela renvoie 48 qui est 3x**2 pour x=4
```

Output

```
In [1]: runfile('C:/Us
WARNING (theano.tensor
48.0
```

Exemple 2 :

```python
from theano import *
import numpy

x = tensor.fvector('x')
target = tensor.fscalar('target')

W = theano.shared(numpy.asarray([0.1, 0.25, 0.15, 0.3]), 'W')
print ("Weights: ", W.get_value())

y = (x * W).sum()
cost = tensor.sqr(target - y)
gradients = tensor.grad(cost, [W])
W_updated = W - (0.1 * gradients[0])
updates = [(W, W_updated)]

f = function([x, target], y, updates=updates)
for i in range(6):
    output = f([1.0, 1.0, 1.0, 1.0], 100.0)
    print ("iteration: ", i)
    print ("Modified Weights: ", W.get_value())
    print ("Output: ", output)
```

Output

```
In [124]: runfile('C:/Users/hajar/Desktop/exercice pytho/exemple/untitled9.py'
wdir='C:/Users/hajar/Desktop/exercice pytho/exemple')
Weights:  [0.1  0.25 0.15 0.3 ]
iteration:  0
Modified Weights:  [19.94 20.09 19.99 20.14]
Output:  0.8
iteration:  1
Modified Weights:  [23.908 24.058 23.958 24.108]
Output:  80.16000000000001
iteration:  2
Modified Weights:  [24.7016 24.8516 24.7516 24.9016]
Output:  96.03200000000001
iteration:  3
Modified Weights:  [24.86032 25.01032 24.91032 25.06032]
Output:  99.2064
iteration:  4
Modified Weights:  [24.892064 25.042064 24.942064 25.092064]
Output:  99.84128
iteration:  5
Modified Weights:  [24.8984128 25.0484128 24.9484128 25.0984128]
Output:  99.968256
iteration:  6
Modified Weights:  [24.89968256 25.04968256 24.94968256 25.0996825
Output:  99.9936512

In [125]:
```

Pretty-print “pp”:

Pour permet de voir comment theano a mis en œuvre une fonction, en utilisant la fonction pretty-print comme suit :

```
1    import theano
2    import theano.tensor as T
3    x = T.dscalar('x')
4    y = x**3
5    output = T.grad(y,x)
6    f = theano.function([x],output)
7    print(f(4))
8    #utilisant la fonction pretty-print (pp)
9    from theano import pp
10   print(pp(output))
```

Output

```
48.0
((fill((x ** TensorConstant{3}), TensorConstant{1.0}) * TensorConstant{3}) * (x ** (TensorConstant{3} - TensorConstant{1})))
```

Ifelse dans theano :

Il s'agit donc de réaliser une ou plusieurs actions en selon la vérification d'une condition.

Syntax :

Var = ifelse(conditionnelle , bloc exécuter si la conditionnelle est vraie, bloc exécuter s

```
1   from theano import tensor as T
2   from theano.ifelse import ifelse
3   import theano, numpy
4   a,b = T.scalars('a','b')
5   x,y = T.matrices('x','y')
6   z_ifelse = ifelse(T.gt(a,b), T.mean(x), T.mean(y))
7   f_ifelse = theano.function([a,b,x,y], z_ifelse)
8   val1 = 0.0
9   val2 = 1.0
10  mat1 = numpy.array([2,2,2],ndmin=2)
11  mat2 = numpy.array([3,3,3],ndmin=2)
12  print("Ifelse : ", f_ifelse(val1, val2, mat1, mat2))
13  print("Ifelse : ", f_ifelse(val2, val1, mat1, mat2))
```

Output

```
Ifelse :  3.0
Ifelse :  2.0
```

La fonction printing:

La fonction print() imprime le message spécifié sur l'écran, ou un autre périphérique de sortie standard. Le message peut être une chaîne, ou tout autre objet, l'objet sera converti en chaîne avant d'être écrit à l'écran.
Exemple :

```python
import theano
from theano import pp, grad
from theano import tensor as tt,function,printing
x = tt.dscalar('x')
y = x ** 2
gy = grad(y, x)
pp(gy)
f = function([x], gy)
print(pp(f.maker.fgraph.outputs[0]))
print(f)
theano.printing.debugprint(gy)
theano.printing.debugprint(f)
```

Output

```
untitled11.py', wdir='C:/Users/hajar/Desktop/exercice pytho/exemple')
(TensorConstant{2.0} * x)
<theano.compile.function_module.Function object at 0x000027E60716610
Elemwise{mul} [id A] ''
 |Elemwise{mul} [id B] ''
 | |Elemwise{second,no_inplace} [id C] ''
 | | |Elemwise{pow,no_inplace} [id D] ''
 | | | |x [id E]
 | | | |TensorConstant{2} [id F]
 | | |TensorConstant{1.0} [id G]
 | |TensorConstant{2} [id F]
 |Elemwise{pow} [id H] ''
   |x [id E]
   |Elemwise{sub} [id I] ''
     |TensorConstant{2} [id F]
     |InplaceDimShuffle{} [id J] ''
       |TensorConstant{1} [id K]
Elemwise{mul,no_inplace} [id A] ''   0
 |TensorConstant{2.0} [id B]
 |x [id C]
```

Conclusion :

Nous avons vu dans ce mini-projet une définition bien détaillée pour la biblio Theano qui est parmi les biblio de python les plus importants, et plus utiliser dans la recherche, particulièrement dans des domaines scientifiques. comme nous avons vu les avantages de cette biblio et finalement nous avons défini certains exemples parmi ses fonctions ...

FIN
nous vous remercions pour votre attention...