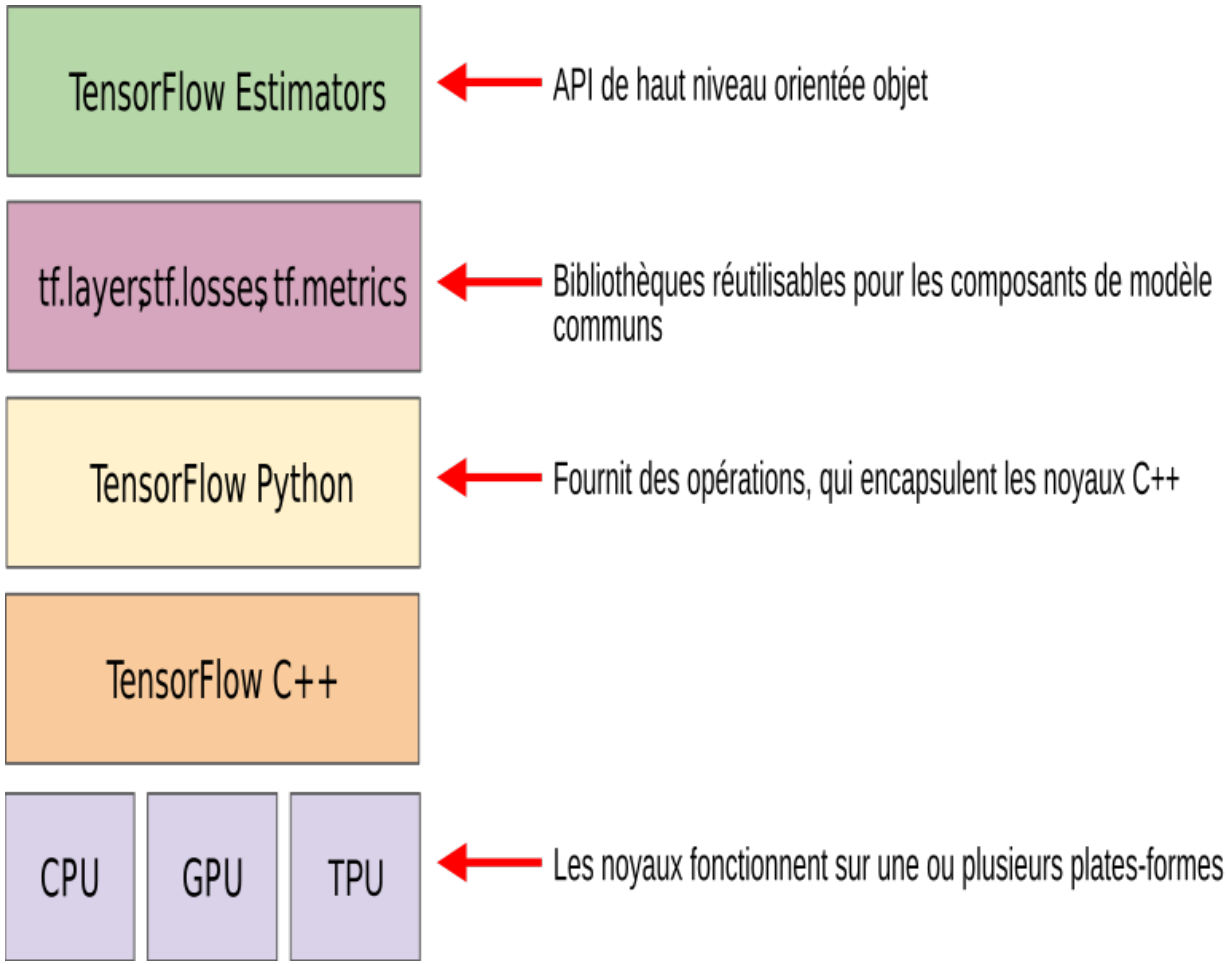
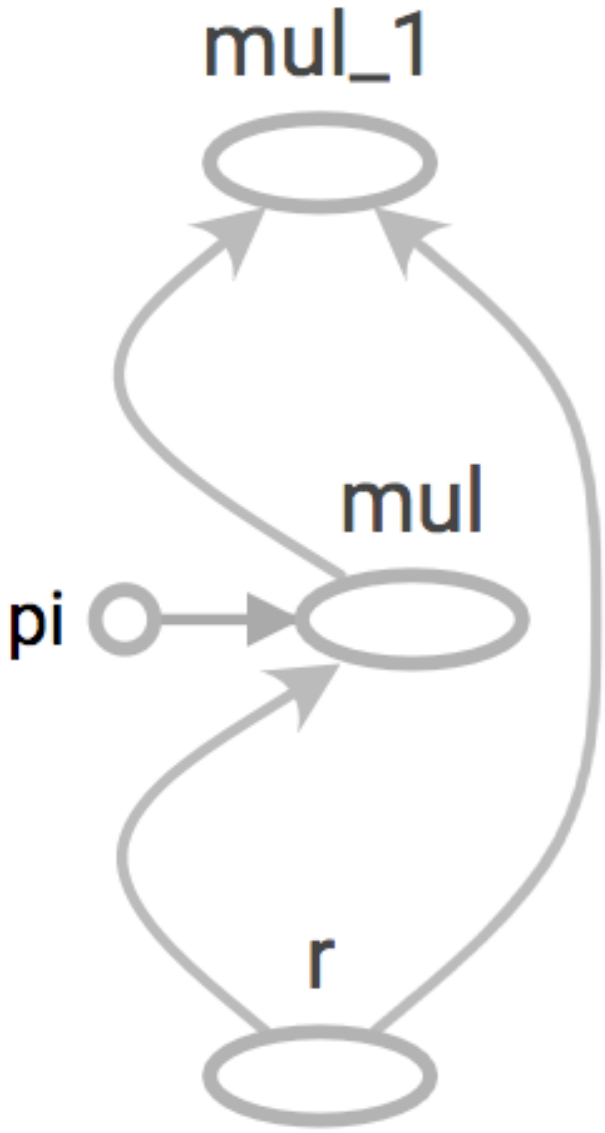
TensorFlow

TensorFlow


TensorFlow Estimators
API de haut niveau orientée objet
tf.layers,tf.losses,tf.metrics
Bibliothèques réutilisables pour les composants de modèle communs
TensorFlow Python
Fournit des opérations, qui encapsulent les noyaux C++
TensorFlow C++
CPU
GPU
TPU
Les noyaux fonctionnent sur une ou plusieurs plates-formes

TensorFlow

$$\pi * r^2 \quad \rightarrow$$



TensorFlow

Créer un graphe :

```
1  import tensorflow as tf
2  a = tf.constant(6)
3  b = tf.constant(7)
4  mul = tf.multiply(a, b, name='a.b')
5  print(mul)
```

```
1  Tensor("a.b:0", shape=(), dtype=int32)
```

TensorFlow

Exécuter un graphe :

```
1 with tf.Session() as session:
2         print(session.run(mul))
```

```
1 42
```

TensorFlow

Créer un graphe tensorflow avec des “placeholder” :

```
1 import tensorflow as tf
2 a = tf.placeholder(tf.float32 , name = "var_a")
3 b = tf.placeholder(tf.float32 , name = "var_b")
4 mul = tf.multiply(a, b, name='a.b')
5 print(mul)
```

```
1 Tensor("a.b:0", dtype=float32)
```

TensorFlow

Executer un graphe avec des “placeholder” :

```
with tf.Session() as session:
    result = session.run(mul, feed_dict={a:[1,2,3], b:[4,5,6]})
    print(result)

```

```
[ 4. 10. 18.]
```

TensorFlow

Matrices et opérations usuelles :

```
1  matrix1 = tf.constant([[1., 2.],[3., 4.]])
2  matrix2 = tf.constant([[5.,6.],[7.,8.]])
3  product = tf.matmul(matrix1, matrix2)
4
5  with tf.Session() as session:
6          result = session.run(product)
7          print(result)
```

```
1  [[19. 22.]
2   [43. 50.]]
```

Régression Linéaire (Définition des variables)

```
1  import tensorflow as tf
2
3  # Paramètres du modèle
4  a = tf.Variable(tf.random_normal((1,1)), dtype=tf.float32,name='a')
5  b = tf.Variable(tf.random_normal((1,1)), dtype=tf.float32,name='b')
6
7  # Entrée et sortie du modèle
8  x = tf.placeholder(tf.float32,name="in")
9  y = tf.placeholder(tf.float32,name="out")
10
11 # Données
12 x_train = [1, 2, 3, 4]
13 y_train = [2.99,5.004,6.98,9.001]
```

Régression Linéaire (Définition de la régression et de son erreur) :

```
# Le modèle de regression linéaire
y_pred=tf.add(tf.multiply(a,x),b,name='y_pred')

# La fonction de perte (Loss)
loss = tf.reduce_sum(tf.square(y_pred - y),name='loss')

# Méthode de recherche du minimum
lr=0.01
train = tf.train.GradientDescentOptimizer(lr).minimize(loss)
```

Régression Linéaire :

```
# Exécution du graphe 1000 fois sur l'ensemble des données
session = tf.Session()
init = tf.global_variables_initializer()
session.run(init) # reset a et b
for i in range(1000):
    session.run(train, {x: x_train, y: y_train})
# Evaluation de l'entrainement du modèle
a_app,b_app,erreur=session.run([a,b,loss],{x:x_train,y:y_train})
print("a: %s b: %s loss: %s"%(a_app, b_app, erreur))
```

```
a: [[2.0009024]] b: [[0.9914936]] loss: 0.00035671948
```

TensorFlow

Enregistrer un graphe de calcul tensorflow :

```
#Sauvegarde de tout le graphe de calcul
sauvegarde=tf.train.Saver()
sauvegarde.save(session,'graphs/mygraph')

!ls -l graphs/
```

```
total 40
-rw-r--r-- 1 root root    71 Oct  8 12:44 checkpoint
-rw-r--r-- 1 root root     8 Oct  8 12:44 mygraph.data-00000-of-00001
-rw-r--r-- 1 root root   150 Oct  8 12:44 mygraph.index
-rw-r--r-- 1 root root 25739 Oct  8 12:44 mygraph.meta
```

Charger un graphe de calcul tensorflow :

```
#Restauration du graphe sauvegardé
savedGraph = tf.train.import_meta_graph('graphs/mygraph.meta')
#session = tf.Session()
savedGraph.restore(session,tf.train.latest_checkpoint('graphs/'))

#Test du modèle avec de nouvelles valeurs
new_x=[5,10,50,100]
new_y_pred = session.run(y_pred,feed_dict={x:new_x})
print("y_pred(",new_x, ") = ", new_y_pred)
```

```
INFO:tensorflow:Restoring parameters from graphs/mygraph
y_pred( [5, 10, 50, 100] ) =
        [[ 10.996005  21.000517 101.03661  201.08173 ]]
```

TensorFlow

Réutiliser le graphe pour ajouter de nouveaux calculs :

```
g = tf.get_default_graph()
y_pred = g.get_tensor_by_name("y_pred:0")
#x = g.get_tensor_by_name("in:0")
# ...

#Construction d'un nouveau calcul à partir de y_pred
new_op = tf.multiply(y_pred, 10)
new_y_pred = session.run(new_op,feed_dict={x:new_x})
print("new_op(",new_x, ") = ", new_y_pred)
```

```
new_op( [5, 10, 50, 100] ) =
        [[ 109.96005  210.00517 1010.36615 2010.8173 ]]
```

TensorFlow

Le code présenté, executable dans colaboratory