TF & KERAS. РОБОТА із ЗОБРАЖЕННЯМИ

Файл: TF_KERAS_Image_01_002

Операції з тензорами. Стандартні функції

Математичні оперції та функції надає модуль: **tf.math**.
Модуль включає:

- Базові арифметичні операції, алгебраічні та тригонометрічні функції
- Функції з комплексними змінними
- Спеціальні математичні функції
- Функції редукції
- Функції сегментації

Математика - повний аналог numpy.

Оперції лінійної алгебри надає модуль **tf.linalg**

Визначення версії TF

```
import tensorflow as tf # імпорт tensofflow
import numpy as np # імпорт numpy
import pprint as pprint # імпорт пакету посиленого друку
print(tf.__version__) # версія TF
```

```
2.15.0
```

Арифметичні операції

Маємо константні тензори

```
scalar1 = tf.constant(2, dtype=tf.int8)
scalar2 = tf.constant(2, shape = (1,1), dtype=tf.int16)
vector1 = tf.constant([1, 2, 3, 4], dtype=tf.float32)
vector2 = tf.constant([10, 20, 30, 40], dtype=tf.float32)
matrix1 = tf.constant([[1, 2, 3, 4], [3, 4, 5, 6], [6, 7, 8, 9]],
dtype=tf.float64)
matrix2 = tf.constant([[1, 2, 3], [3, 4, 5], [6, 7, 8], [7, 8, 9]],
dtype=tf.float64)
```

Додавання (покомпонентне)

```
add_tensor_1 = tf.add(vector1,vector2)
print('Value-->', add_tensor_1.numpy())
add_tensor_2 = vector1 + vector2
print('Value-->', add_tensor_2.numpy())
add_tensor_22 = vector1 + 5
print('Value-->', add_tensor_22.numpy())
```

```
Value--> [11. 22. 33. 44.]
Value--> [11. 22. 33. 44.]
Value--> [6. 7. 8. 9.]
```

```
scalar_x = tf.constant(5, dtype=tf.float32)
add_tensor_222 = vector1 + scalar_x
print('Value-->', add_tensor_222.numpy())
```

```
Value--> [6. 7. 8. 9.]
```

Віднімання (покомпонентне)

```
sub_tensor_1 = tf.subtract(vector1,vector2)
print('Value-->', sub_tensor_1.numpy())
sub_tensor_2 = vector1 - vector2
print('Value-->', sub_tensor_1.numpy())
# sub_tensor_3 = vector1 - scalar2
sub_tensor_3 = vector1 - np.float32(scalar2)
print('Value-->', sub_tensor_3.numpy())
```

```
Value--> [ -9. -18. -27. -36.]
Value--> [ -9. -18. -27. -36.]
Value--> [[-1.  0.  1.  2.]]
```

Множення скаляр X тензор [tf.math.scalar_mul]

tf.math.scalar_mul(scalar, x, name=None)

```
mul_1 = tf.scalar_mul(np.float32(scalar1), vector2)
# mul_1 = tf.scalar_mul(np.float32(scalar2), vector2) #НЕ РАБОТАЕТ.
print('a    -->', scalar1.numpy())
print('b    -->', vector2.numpy())
print(mul_1)
print('Value-->', mul_1.numpy())

mul_11  =  vector2 * 2
print(mul_11)
print('Value-->', mul_11.numpy())
```

```
a    --> 2
b    --> [10. 20. 30. 40.]
tf.Tensor([20. 40. 60. 80.], shape=(4,), dtype=float32)
Value--> [20. 40. 60. 80.]
tf.Tensor([20. 40. 60. 80.], shape=(4,), dtype=float32)
Value--> [20. 40. 60. 80.]
```

Множення тензор Х тензор (покомпонентне) **tf.math.multiply**

tf.math.multiply(x, y, name=None)

```
mul_2 = tf.multiply(vector1, vector2)
mul_3 = vector1 * vector2
print('a    -->', vector1.numpy())
print('b    -->', vector2.numpy())
print('Value-->', mul_2.numpy())
print('Value-->', mul_3.numpy())
```

```
a    --> [1. 2. 3. 4.]
b    --> [10. 20. 30. 40.]
Value--> [ 10.  40.  90. 160.]
Value--> [ 10.  40.  90. 160.]
```

Ділення тензор на тензор (покомпонентно)

tf.math.divide(x, y, name=None)

```
div_2 = tf.divide(vector2, vector1)
div_3 = vector2 / vector1 # ПОКОМПОНЕНТНО
div_4 = vector2 / vector1[1]  # НА СКАЛЯР
print('a    -->', vector1.numpy())
print('b    -->', vector2.numpy())
print('Value-->', div_2.numpy())
print('Value-->', div_3.numpy())
print('Value-->', div_4.numpy())
```

```
a    --> [1. 2. 3. 4.]
b    --> [10. 20. 30. 40.]
Value--> [10. 10. 10. 10.]
Value--> [10. 10. 10. 10.]
Value--> [ 5. 10. 15. 20.]
```

Звуження тензорів dotproduct

Звуження тензорів вдовж визначеним осям **tf.tensordot**

tf.tensordot(a, b, axes, name=None)

Tensordot підсумовує добуток елементів з a і b за індексами, заданими axes.
https://www.tensorflow.org/api_docs/python/tf/tensordot

AXES:
Either a scalar N, or a list or an int32 Tensor of shape [2, k]. If axes is a scalar, sum over the last N axes of a and the first N axes of b in order. If axes is a list or Tensor the first and second row contain the set of unique integers specifying axes along which the contraction is computed, for a and b, respectively. The number of axes for a and b must be equal. If axes=0, computes the outer product between a and b.

Вектора (тензор рангу 1)

```
Vec_A = tf.constant([1, 2, 3], tf.int32)
Vec_B = tf.constant([10, 20, 30], tf.int32)
print('Vec_A -->', Vec_A.numpy(), 'Ранг -->', tf.rank(Vec_A).numpy())
print('Vec_B -->', Vec_B.numpy(), 'Ранг -->', tf.rank(Vec_B).numpy())
```

```
Vec_A --> [1 2 3] Ранг --> 1
Vec_B --> [10 20 30] Ранг --> 1
```

Приклад 1.0: Поелементне множення двох векторів

```
vec_mul_ = tf.multiply(Vec_A, Vec_B)
print('Dot axis 0 -->', vec_mul_.numpy())
```

```
Dot axis 0 --> [10 40 90]
```

Приклад 1.1: "Векторне" множення двох векторів (зовнішній добуток)

```
# "Векторне" множення
vec_dot_0 = tf.tensordot(Vec_A, Vec_B, axes=0)
print('Dot axis 0 -->', vec_dot_0.numpy(), 'Ранг -->',
tf.rank(vec_dot_0).numpy())
```

```
Dot axis 0 --> [[10 20 30]
 [20 40 60]
 [30 60 90]] Ранг --> 2
```

Приклад 1.2: Скалярне множення двох векторів

```
# Скалярне множення
vec_dot_1 = tf.tensordot(Vec_A, Vec_B, axes=1)
print('Dot axis 1 -->', vec_dot_1.numpy(), 'Ранг -->',
tf.rank(vec_dot_1).numpy())
```

```
Dot axis 1 --> 140 Ранг --> 0
```

2. Матриця

```
# Маємо тензори рангу 2
# A = tf.random.uniform([3, 3], 0, 10, tf.int32)
# B = tf.random.uniform([3, 3], 10, 20, tf.int32)
A = tf.constant([[1, 2, 3],[4, 5, 6],[7, 8, 9]], tf.int32)
B = tf.constant([[10, 20, 30],[40, 50, 60],[70, 80, 90]], tf.int32)
print('A Value-->', A.numpy())
print('B Value-->', B.numpy())
```

```
A Value--> [[1 2 3]
 [4 5 6]
 [7 8 9]]
B Value--> [[10 20 30]
 [40 50 60]
 [70 80 90]]
```

Приклад 2.1: коли A і B є матрицями рангу 2 та axes = 0 дає зовнішнє множення, повретається тензор порядку 4.

```
tdot_1 = tf.tensordot(A, B, axes=0)
print('Dot Shape -->', tf.shape(tdot_1).numpy())
print('Value-->', tdot_1.numpy())
print('Value--> 1,1,1,1', tdot_1[1,1,1,1].numpy())
```

```
Dot Shape --> [3 3 3 3]
Value--> [[[[ 10  20  30]
   [ 40  50  60]
   [ 70  80  90]]

  [[ 20  40  60]
   [ 80 100 120]
   [140 160 180]]

  [[ 30  60  90]
   [120 150 180]
   [210 240 270]]]
```

```
 [[[ 40  80 120]
   [160 200 240]
   [280 320 360]]

  [[ 50 100 150]
   [200 250 300]
   [350 400 450]]

  [[ 60 120 180]
   [240 300 360]
   [420 480 540]]]
```

```
 [[[ 70 140 210]
   [280 350 420]
   [490 560 630]]

  [[ 80 160 240]
   [320 400 480]
   [560 640 720]]

  [[ 90 180 270]
   [360 450 540]
   [630 720 810]]]]
Value--> 1,1,1,1 250
```

Приклад 2.2: коли A і B є матрицями рангу 2 та axes = 1 внутрішнє (перекрестне) множення матриці A на B.

```
tdot_1 = tf.tensordot(A, B, axes=1)
print('Dot Shape -->', tf.shape(tdot_1).numpy())
print('Value-->', tdot_1.numpy())
```

```
Dot Shape --> [3 3]
Value--> [[ 300  360  420]
 [ 660  810  960]
 [1020 1260 1500]]
```

```
#Перевірка
print('Result', (A[1,:]*B[:,1]).numpy())
print('Sum', tf.reduce_sum(A[1,:]*B[:,1]).numpy())
```

```
Result [ 80 250 480]
Sum 810
```

Приклад 2.3: коли A і B є матрицями рангу 2 та axes = [[0], [1]] внутрішнє (перекрестне) множення матриці A на B.

```
tdot_2 = tf.tensordot(A, B, axes=[[0],[1]])
print('Dot Shape -->', tf.shape(tdot_2).numpy())
print('Value-->', tdot_2.numpy())
```

```
Dot Shape --> [3 3]
Value--> [[ 300  660 1020]
 [ 360  810 1260]
 [ 420  960 1500]]
```

```
#Перевірка
print('Result', (A[:,0]*B[:,1]).numpy())
print('Sum', tf.reduce_sum(A[:,0]*B[:1]).numpy())
```

```
Result [ 20 200 560]
Sum 300
```

Приклад 2.3 : коли A і B є матрицями рангу 2 та axes = [[1], [0]] внутрішнє (перекрестне) множення матриці B на A.

```
tdot_3 = tf.tensordot(A, B, axes=[[1],[0]])
print('Dot Shape -->', tf.shape(tdot_3).numpy())
print('Value-->', tdot_3.numpy())
```

```
Dot Shape --> [3 3]
Value--> [[ 300  360  420]
 [ 660  810  960]
 [1020 1260 1500]]
```

```
tdot_4 = tf.tensordot(A, B, axes=0)
print('Dot Rank -->', tf.rank(tdot_4).numpy())
print('Dot Shape -->', tf.shape(tdot_4).numpy())
print('Value-->', tdot_4.numpy())
```

```
Dot Rank --> 4
Dot Shape --> [3 3 3 3]
Value--> [[[[ 10  20  30]
   [ 40  50  60]
   [ 70  80  90]]

  [[ 20  40  60]
   [ 80 100 120]
   [140 160 180]]

  [[ 30  60  90]
   [120 150 180]
   [210 240 270]]]
```

```
[[[ 40  80 120]
  [160 200 240]
  [280 320 360]]

 [[ 50 100 150]
  [200 250 300]
  [350 400 450]]

 [[ 60 120 180]
  [240 300 360]
  [420 480 540]]]
```

```
[[[ 70 140 210]
  [280 350 420]
  [490 560 630]]

 [[ 80 160 240]
  [320 400 480]
  [560 640 720]]

 [[ 90 180 270]
  [360 450 540]
  [630 720 810]]]]
```

3-вимірний тензор

Приклад 4: Припустимо, що $A_{i,j,k}$ та $B_{l,m,n}$ представляють два тензори рангу 3. Тоді dot(A, B, [[0], [2]]) - це тензор C_{jklm} рангу 4, запис якого, що відповідає індексам (j, k, l, m), задається наступним чином:
$C_{jklm} = \sum_i a_{ijk} b_{lmi}$.

Загалом order(C) = order(A) + order(B) - 2*len(axes[0]).

axes: Може бути або скаляр N, або список, або тензор типу int32[2, k]. Якщо axes є скаляром, підсумуйте останні N осей A та перші N осі B по порядку. Якщо axes — це список або тензор, перший і другий рядки містять набір унікальних цілих чисел, що визначають осі, уздовж яких обчислюється скорочення, для A і B відповідно. Кількість осей для A і B має бути рівною. Якщо axes=0, обчислюється зовнішній добуток між A і B.

```
C = tf.constant([[[1,2,3],[4,5,6],[7,8,9]],
                 [[11,12,13],[14,15,16],[17,18,19]],
                 [[21,22,23],[24,25,26],[27,28,29]]], dtype=tf.int32)
D = tf.constant([[[10,20,30],[40,50,60],[70,80,90]],
                 [[110,120,130],[140,150,160],[170,180,190]],
                 [[210,220,230],[240,250,260],[270,280,290]]], dtype=tf.int32)
print('Ранг C -->', tf.rank(C).numpy())
print('Ранг D -->', tf.rank(D).numpy())
print('C Value-->', C.numpy())
print('D Value-->', D.numpy())
```

```
Ранг C --> 3
Ранг D --> 3
C Value--> [[[ 1  2  3]
  [ 4  5  6]
  [ 7  8  9]]

 [[11 12 13]
  [14 15 16]
  [17 18 19]]

 [[21 22 23]
  [24 25 26]
  [27 28 29]]]
D Value--> [[[ 10  20  30]
  [ 40  50  60]
  [ 70  80  90]]

 [[110 120 130]
  [140 150 160]
  [170 180 190]]

 [[210 220 230]
  [240 250 260]
  [270 280 290]]]
```

```
tdot_5 = tf.tensordot(C, D, axes=0)
print('Ранг Dot -->', tf.rank(tdot_5).numpy())
print(tdot_5[:,0,0,:,:,:].numpy())
```

```
Ранг Dot --> 6
[[[[  10   20   30]
   [  40   50   60]
   [  70   80   90]]

  [[ 110  120  130]
   [ 140  150  160]
   [ 170  180  190]]

  [[ 210  220  230]
   [ 240  250  260]
   [ 270  280  290]]]
```

```
[[[ 110  220  330]
  [ 440  550  660]
  [ 770  880  990]]

 [[1210 1320 1430]
  [1540 1650 1760]
  [1870 1980 2090]]

 [[2310 2420 2530]
  [2640 2750 2860]
  [2970 3080 3190]]]
```

```
[[[ 210  420  630]
  [ 840 1050 1260]
  [1470 1680 1890]]

 [[2310 2520 2730]
  [2940 3150 3360]
  [3570 3780 3990]]

 [[4410 4620 4830]
  [5040 5250 5460]
  [5670 5880 6090]]]]
```

```
tdot_5 = tf.tensordot(C, D, axes=1)
print('Ранг Dot -->', tf.rank(tdot_5).numpy())
print(tdot_5[:,0].numpy())
```

```
Ранг Dot --> 4
[[[  860   920   980]
  [ 1040  1100  1160]
  [ 1220  1280  1340]]

 [[ 4160  4520  4880]
  [ 5240  5600  5960]
  [ 6320  6680  7040]]

 [[ 7460  8120  8780]
  [ 9440 10100 10760]
  [11420 12080 12740]]]
```

```
tdot_51 = tf.tensordot(C, D, axes=0)
tdot_52 = tf.tensordot(C, D, axes=1)
tdot_53 = tf.tensordot(C, D, axes=((0),(1)))
tdot_54 = tf.tensordot(C, D, axes=((0),(2)))
tdot_55 = tf.tensordot(C, D, axes=((1),(2)))
print('Ранг Tdot 51-->', tf.rank(tdot_51).numpy())
print('Ранг Tdot 52-->', tf.rank(tdot_52).numpy())
print('Ранг Tdot 53-->', tf.rank(tdot_53).numpy())
print('Ранг Tdot 54-->', tf.rank(tdot_54).numpy())
print('Ранг Tdot 54-->', tf.rank(tdot_55).numpy())
```

```
Ранг Tdot 51--> 6
Ранг Tdot 52--> 4
Ранг Tdot 53--> 4
Ранг Tdot 54--> 4
Ранг Tdot 54--> 4
```

```
print('52 -->  ', tdot_52[0,1,2,1].numpy())
print('53 -->  ', tdot_53[0,1,2,1].numpy())
print('54 -->  ', tdot_54[0,1,2,1].numpy())
print('55 -->  ', tdot_55[0,1,2,1].numpy())
```

```
52 -->   2900
53 -->   9600
54 -->   9200
55 -->   3810
```

Додаткові операції з тензорами

- div(x, y, name=None) Divides the elements of two tensors
- add_n(inputs, name=None) Adds multiple tensors
- floormod(x, y, name=None) Performs the modulo operation
- abs(x, name=None) Computes the absolute value
- negative(x, name=None) Negates the tensor's elements
- sign(x, name=None) Extracts the signs of the tensor’s element
- reciprocal(x, name=None) Computes the reciprocals

Приклади операцій (функцій)

```
Vec_C = tf.random.uniform([5], -10, 10, tf.float32)
print('Vec_C -->', Vec_C.numpy())
sig_C = tf.sign(Vec_C)
print('Sig_C -->', sig_C.numpy())
abs_C = tf.abs(Vec_C)
print('Abs_C -->', abs_C.numpy())
mod_C = tf.math.floormod(Vec_C, 3) # remainder of division
print('Mod_C -->', mod_C.numpy())
```

```
Vec_C --> [ 5.7243233 -0.887907  -6.3002324  9.856071   7.6254654]
Sig_C --> [ 1. -1. -1.  1.  1.]
Abs_C --> [5.7243233 0.887907  6.3002324 9.856071  7.6254654]
Mod_C --> [2.7243233 2.112093  2.6997676 0.8560715 1.6254654]
```

Операції (функції) типу REDUCE

- tf.reduce_sum(m) сума елементів тензору
- tf.reduce_min(m) мінімальне
- tf.reduce_max(m) максимальне
- tf.reduce_mean(m) середнє

Приклади

```
tens_A = tf.random.uniform([10], 0, 100, tf.int32)
print('Tensor A -->', tens_A.numpy())
sum_A = tf.reduce_sum(tens_A)
print('Sum A -->', sum_A.numpy())
min_A = tf.reduce_min(tens_A)
print('Min A -->', min_A.numpy())
ind_min = tf.argmin(tens_A)
print('Idex min A -->', ind_min.numpy())
max_A = tf.reduce_max(tens_A)
print('Max A -->', max_A.numpy())
ind_max = tf.argmax(tens_A)
print('Idex max A -->', ind_max.numpy())
mean_A = tf.reduce_mean(tens_A)
print('Mean A -->', mean_A.numpy())
```

```
Tensor A --> [91 57 97 86 92 76 96 83 51 59]
Sum A --> 788
Min A --> 51
Idex min A --> 8
Max A --> 97
Idex max A --> 2
Mean A --> 78
```

```
tens_B = tf.random.uniform([5,5], 0, 100, tf.int32)
```

```
print('Tensor B -->', tens_B.numpy())
sum_B = tf.reduce_sum(tens_B)
print('Sum B -->', sum_B.numpy())
min_B = tf.reduce_min(tens_B)
print('Min B-->', min_B.numpy())
ind_min_ax0 = tf.argmin(tens_B, axis = 0)
print('Idex min B ax0 -->', ind_min_ax0.numpy())
ind_min_ax1 = tf.argmin(tens_B, axis = 1)
print('Idex min B ax1 -->', ind_min_ax1.numpy())

max_B = tf.reduce_max(tens_B)
print('Max B -->', max_B.numpy())
ind_max = tf.argmax(tens_B)
print('Idex max B -->', ind_max.numpy())
mean_B = tf.reduce_mean(tens_B)
print('Mean B -->', mean_B.numpy())
```

```
Tensor B --> [[65 80 52 50 46]
 [49  1 61 85 31]
 [67 61 45  0 66]
 [66 32 52 99 11]
 [69 83 56 44 59]]
Sum B --> 1330
Min B--> 0
Idex min B ax0 --> [1 1 2 2 3]
Idex min B ax1 --> [4 1 3 4 3]
Max B --> 99
Idex max B --> [4 4 1 3 2]
Mean B --> 53
```