1. **Understanding output for Conv2D model summary**

from tensorflow import keras

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Flatten,Conv2D

model = Sequential()

model.add(Conv2D(32,kernel\_size=(6,5),strides=(3, 4),input\_shape=(28,28,2)))

#(6\*5\*2+1)\*32=1952 ; '1' added for bias

#[(28-6)+1]/3 ~ 8 ; '1' added for bias

#[(28-5)+1]/4 =6 ; '1' added for bias

model.add(Flatten())

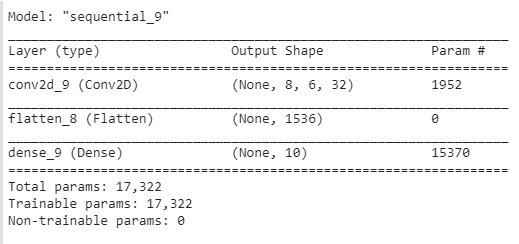
#(8\*6\*32 = 1536)

model.add(Dense(10,activation='relu'))

#[(1536\*10)+10=15370]; ‘10’ added as bias for 10 neurons declared in #Dense layer

model.summary()

OUTPUT:



1. **Uploading .csv file in Colab and access it after converting to numpy array**

import os

import numpy as np

import pandas as pd

from tensorflow import split, shape, reshape, size

path = "/content/sample\_data/sampletest.csv"

if os.path.exists(path):

  df = pd.read\_csv(path)

  print(f"Got file {path}")

  print(df)

else:

    print(f"Unable to find the file at {path}")

z = np.array(df, dtype='float32')

print(f"Shape for z ->", shape(z))

x = z[1,0:5:2]

print(f"Output for x ->", x)

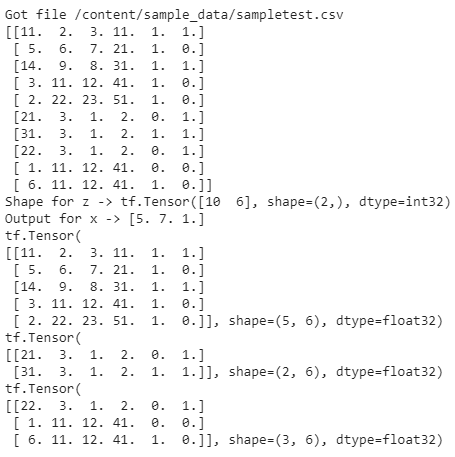
var1,var2,var3 = split(z,[5,2,3],0)

print(var1)

print(var2)

print(var3)

OUTPUT:



The output specifies tensor of [10,6] representing 10 rows and 6 columns.

Shape is (2,) specifying that tensor is a ‘vector’ having 2 values 10 & 6.

z[1,0:5:2] >> to access values starting at row 1,

column starting at ‘0’,

consider upto ‘5’ elements,

‘2nd’ element considered from previous value.

Thereby output of x being ‘[5 7 1]’

The ‘split’ function splits the tensor z into three variables var1, var2, var3 with lengths 5,2,3.

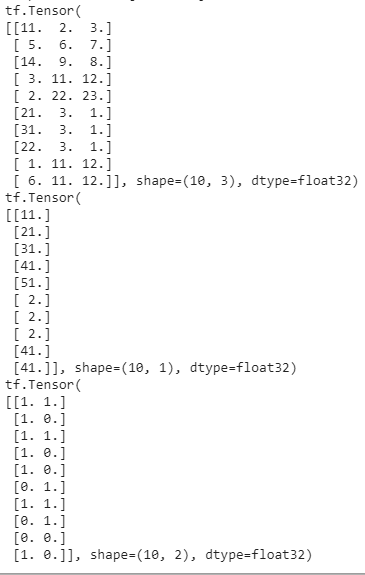
Since we have given the third parameter axis=0, the split will happen considering the number of rows in the tensor. In this case the number of rows is ‘10’ and the lengths of var1, var2, var3 should add upto ‘10’ (5+2+3=10).

If we specify the third parameter axis=1, then split will happen considering the number of columns in the tensor. In this case the number of columns is ‘6’.

If we specify the code as >> var1,var2,var3 = split(z,[3,1,2],1)

Then the 3 variables will be of lengths 3,1,2 (which when added will be ‘6’ corresponding to number of columns).

Output for this split would be-



1. **Understanding ‘reshape’**

In addition to above code, on adding the below code of reshape; the output generates a tensor with the modified shape-

zvar1 = reshape(var1,(size(var1),))

print(f"Shape of zvar1 ->", zvar1)

zvar2 = reshape(var2,(size(var2),))

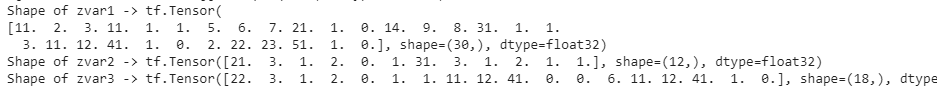
zvar3 = reshape(var3,(18,))

#if appropriate matching size is not provided, system will throw error

print(f"Shape of zvar2 ->",zvar2)

print(f"Shape of zvar3 ->",zvar3)

OUTPUT:



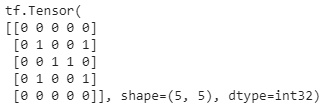
1. **Understanding ‘Confusion Matrix’**

from tensorflow.math import confusion\_matrix

conf = confusion\_matrix([2,2,1,1,3,3],[2,3,4,1,1,4])

print(conf)

OUTPUT:



The maximum number in the vectors of confusion matrix is ‘4’, so a zero matrix will be created with shape (5,5) i.e. 4+1 = 5.

‘1’ will be plotted in the zero-matrix taking the position from the values present in the vectors. In our case the position of ‘1’ is as below-

(2,2), (2,3), (1,4), (1,1), (3,1), (3,4)

1. **Understanding ‘Transpose’ and its parameter ‘permute’**

from tensorflow import transpose

v1 = [[[1 ,2 ,3,22],[82, 13, 14,11]],

      [[7, 8, 9,21],[12, 13, 14,22]],

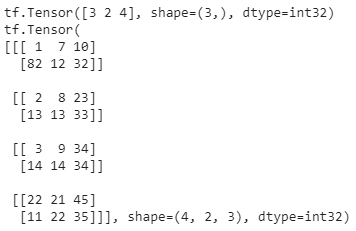
      [[10,23,34,45],[32,33,34,35]]]

print(shape(v1))

tran=transpose(v1)

print(tran)

OUTPUT:



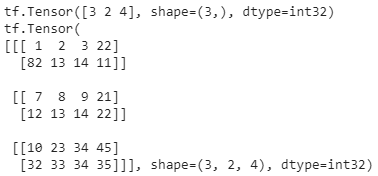
We can observe that the shape of v1 is [3,2,4] and after transpose the shape is changed to [4,2,3].

We can control the shape of the transpose by using its permutation parameter ‘perm’.

On giving the code as-

tran=transpose(v1, perm=[0,1,2])

Output is-



Shape of v1 is [3,2,4], by having perm=[0,1,2];

3 -> is at position 0

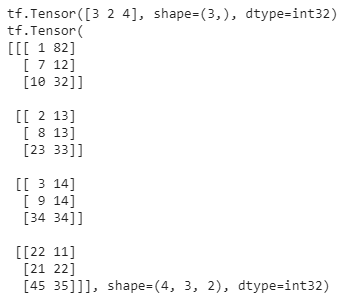
2 -> is at position 1

4 -> is at position 2

it forces the shape of the transpose by controlling the position of the elements in shape v1.

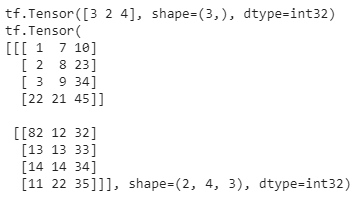
e.g. if perm=[2,0,1] then transpose shape would be [4,3,2]

Output is-



if perm=[1,2,0] then transpose shape would be [2,4,3]

Output is-



1. **Understanding ‘concat’**

from tensorflow import concat

t1= [[[1, 2], [2, 3]], [[4, 4], [5, 3]]]

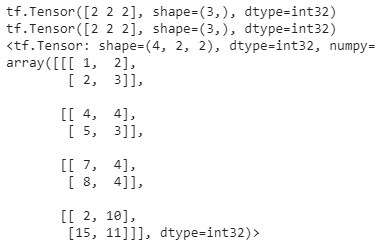
t2=[[[7, 4], [8, 4]], [[2, 10], [15, 11]]]

print(shape(t1))

print(shape(t2))

concat([t1, t2],0)

OUTPUT:



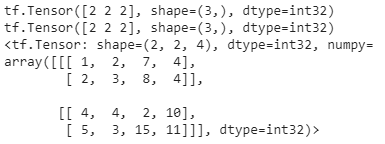
Here shape of t1 is [2,2,2] and shape of t2 is [2,2,2].

‘concat’ concatenates t1 & t2 at axis ‘0’; so the 0th position of the vectors t1 & t2 are added and the resultant shape of concatenation is [4,2,2] and the resultant tensor is been created accordingly.

Here concatenating dimensions can be provided in the range [-3,3).

The axis parameter can have values -3, -2, -1, 0, 1, 2. #value 3 is out of range

If axis is given value ‘2’, the output will be-



Thereby adding the values on the 2nd position of the vectors t1 and t2 to get the resultant shape (2,2,4).

The resultant shape for axis ‘-3’ is same as for axis ‘0’

The resultant shape for axis ‘-2’ is same as for axis ‘1’

The resultant shape for axis ‘-1’ is same as for axis ‘2’

1. **Understanding ‘stack’**

from tensorflow import stack

t1 = [[1, 2,0,11]]

t2 = [[7, 4,3,12]]

t3 = [[9,8,10,13]]

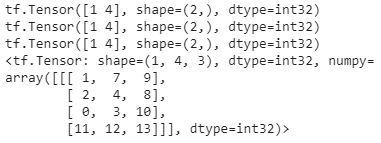
print(shape(t1))

print(shape(t2))

print(shape(t3))

stack([t1,t2,t3],2)

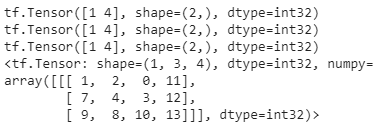
OUTPUT:



Here we have 3 vectors t1, t2, t3 in which ‘stack’ is used, so in the resultant shape ‘3’ is included in the position depending on the axis value provided in the stack function.

In our case axis=2; so the resultant shape is (1,4,3)

If axis=1, then resultant shape is (1,3,4) and the output is-



The axis parameter can have values -3, -2, -1, 0, 1, 2. #value 3 is out of range.