Practical Python For Tensorflow

Tony Xu

10/7/2017

Agenda

- **Recap of what we have learned last time**
- Python Solution
- C/C++ Solution
- TensorFlow Solution

Practice on Convolution

- **Assuming stride = 1; height and width are same for input, output, kernel.**
- **Q: if we want output size equals to the input size. What we need to do to the input?**
- **->padding**
- **What’s the size for padding, if kernel size = k?**
- **->padding size = (k-1)/2 , k=kernel size**
- **More general formula :** $O = (I - k + 2p)/s + 1$
  - **O is output size**
  - **I is input size**
  - **k is kernel size**
  - **p is padding size**
  - **s is stride size, usually is 1**

Agenda

- Recap of what we have learned last time
- **Python Solution**
- C/C++ Solution
- TensorFlow Solution

Python Solution

- **We only use Python primitives to illustrate how to program in Python, but you could use Numpy (Numerical Python)…**
- **We are going to show you step by step with a few small examples before we form a solution.**

More on Python List

- Initialization of a short list
- >>> a = [0, 0, 0]
- How about a long list …100 items? 1000?
- >>> a = [0 for i in range(100)]
- How about multiple dimensions?
- >>> a = [[0 for in range(10)] for j in range(5)]

Rows

Cols

Demo

Agenda

- Recap of what we have learned last time
- Python Solution
- **C/C++ Solution**
- TensorFlow Solution

C/C++ Solution

- Assume you know C/C++ already
- The challenge/difficulty is to access a 2D array with pointers

Demo

Agenda

- Recap of what we have learned last time
- Python Solution
- C/C++ Solution
- **TensorFlow Solution**

Quiz

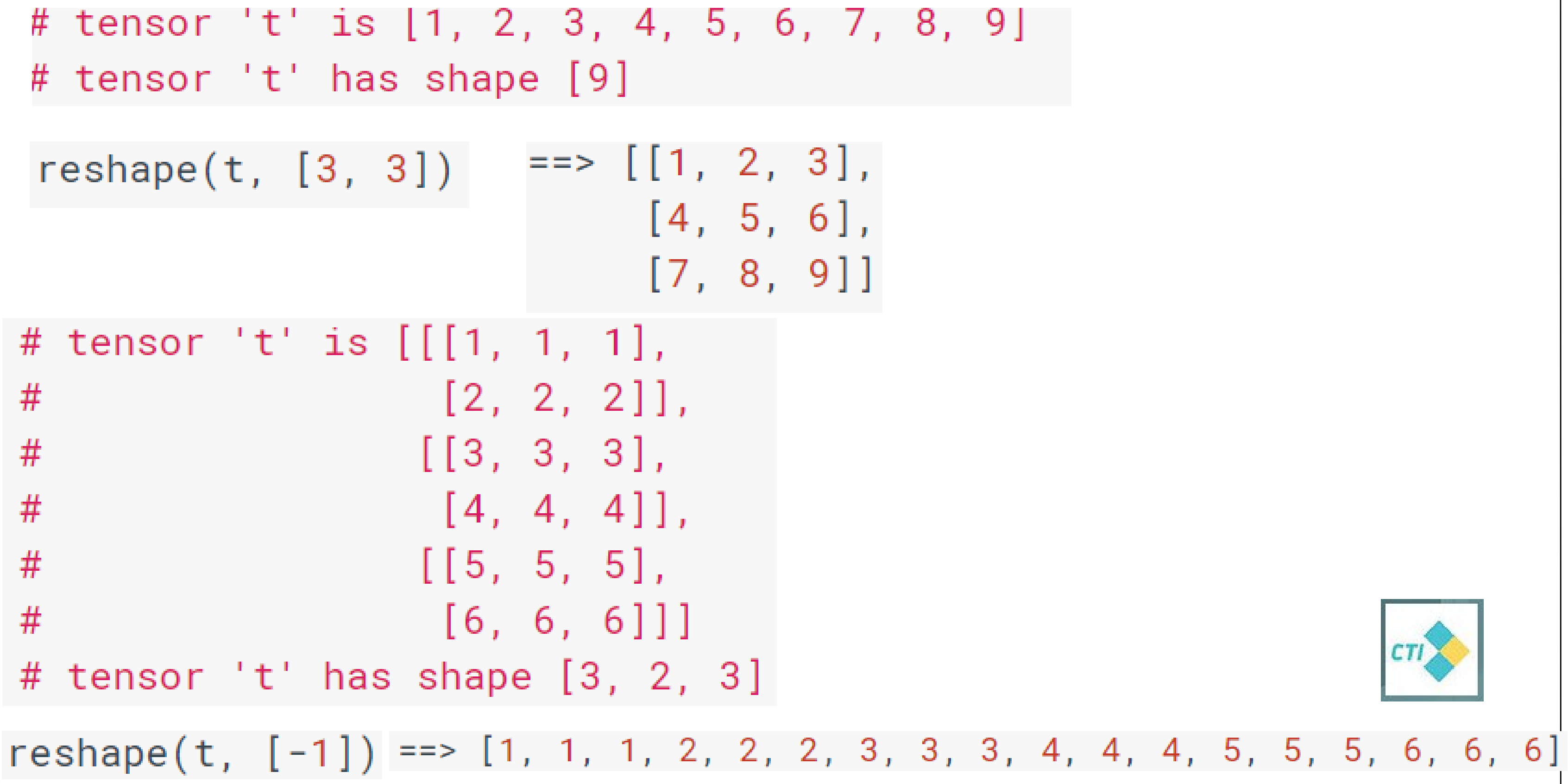
- **How many sections does a TensorFlow program usually have?**
- **2**
- **What are they?**
- **Graph and Session**
- **Why do we divide it to two sections?**
- **Reduce the overhead, easily distribute jobs to different devices.**

TensorFlow Solution

- Introduce a few TensorFlow functions.
- Practices in Convolution
- We also show you a small step each time and finally reach to our goal.

Tensor Shape and Reshape

- **We always need to manipulate tensor's shape to meet desired math**



```
# tensor 't' is [1, 2, 3, 4, 5, 6, 7, 8, 9]
# tensor 't' has shape [9]
```

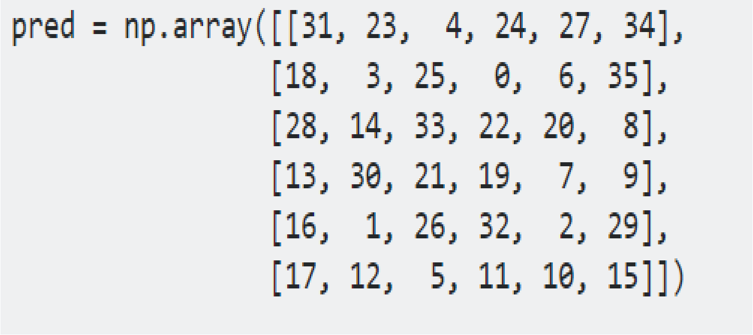
```
reshape(t, [3, 3])    ==> [[1, 2, 3],
                           [4, 5, 6],
                           [7, 8, 9]]
```

```
# tensor 't' is [[[1, 1, 1],
#                 [2, 2, 2]],
#                [[3, 3, 3],
#                 [4, 4, 4]],
#                [[5, 5, 5],
#                 [6, 6, 6]]]
# tensor 't' has shape [3, 2, 3]
```

```
reshape(t, [-1]) ==> [1, 1, 1, 2, 2, 2, 3, 3, 3, 4, 4, 4, 5, 5, 5, 6, 6, 6]
```

TensorFlow: argmax

- **tf.argmax:Returns the index with the largest value across axes of a tensor**

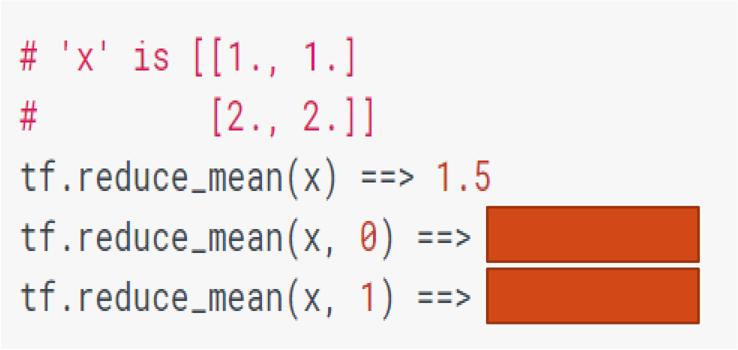


```
pred = np.array([[31, 23,  4, 24, 27, 34],
                [18,  3, 25,  0,  6, 35],
                [28, 14, 33, 22, 20,  8],
                [13, 30, 21, 19,  7,  9],
                [16,  1, 26, 32,  2, 29],
                [17, 12,  5, 11, 10, 15]])
```

- >>> **argmax(pred, 1)**
- □*array([5, 5, 2, 1, 3, 0])*
- **Remember: 0: vertical, 1: horizontal**

TensorFlow: reduce mean

- **tf.reduce_mean: Computes the mean of elements across dimensions of a tensor.**

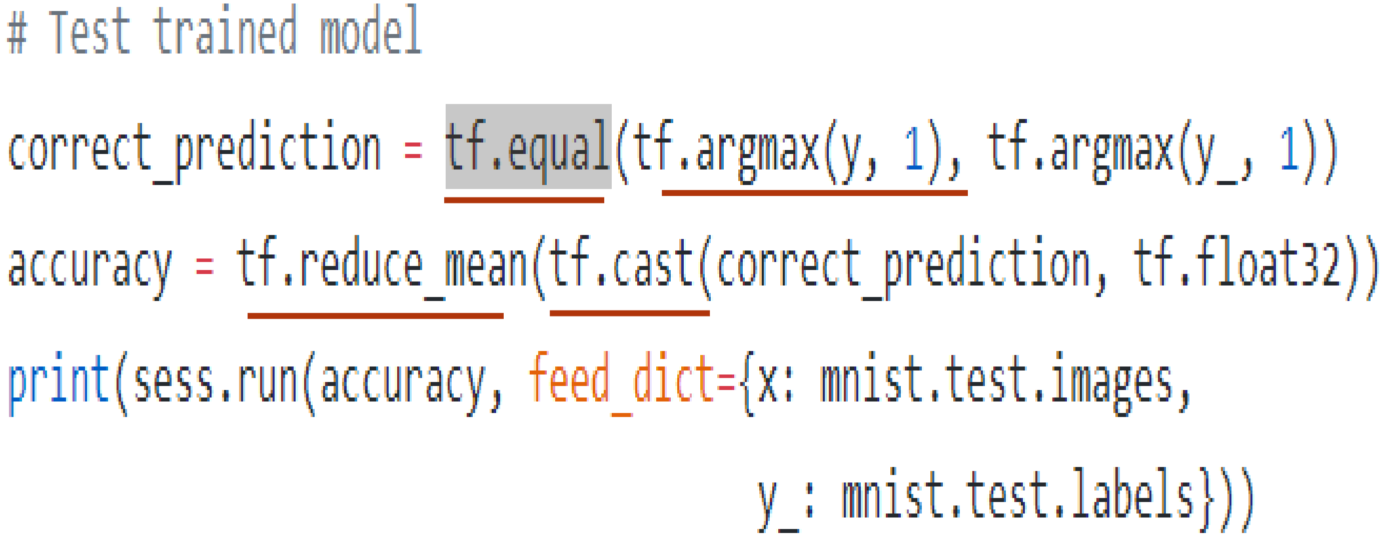


```
# 'x' is [[1., 1.]
#         [2., 2.]]
tf.reduce_mean(x) ==> 1.5
tf.reduce_mean(x, 0) ==> [redacted]
tf.reduce_mean(x, 1) ==> [redacted]
```

TensorFlow: Equal

- tf.equal(x,y): Returns the truth value of (x == y) element-wise. (Returns a true-false tensor)
- x, y are tensors with same shape.
- If we have x = tf.constant([1,2,3]), then tf.equal(x,x) will give us:
- [true, true, true]
- tf.not_equal, tf.less etc. are in similar fashion.

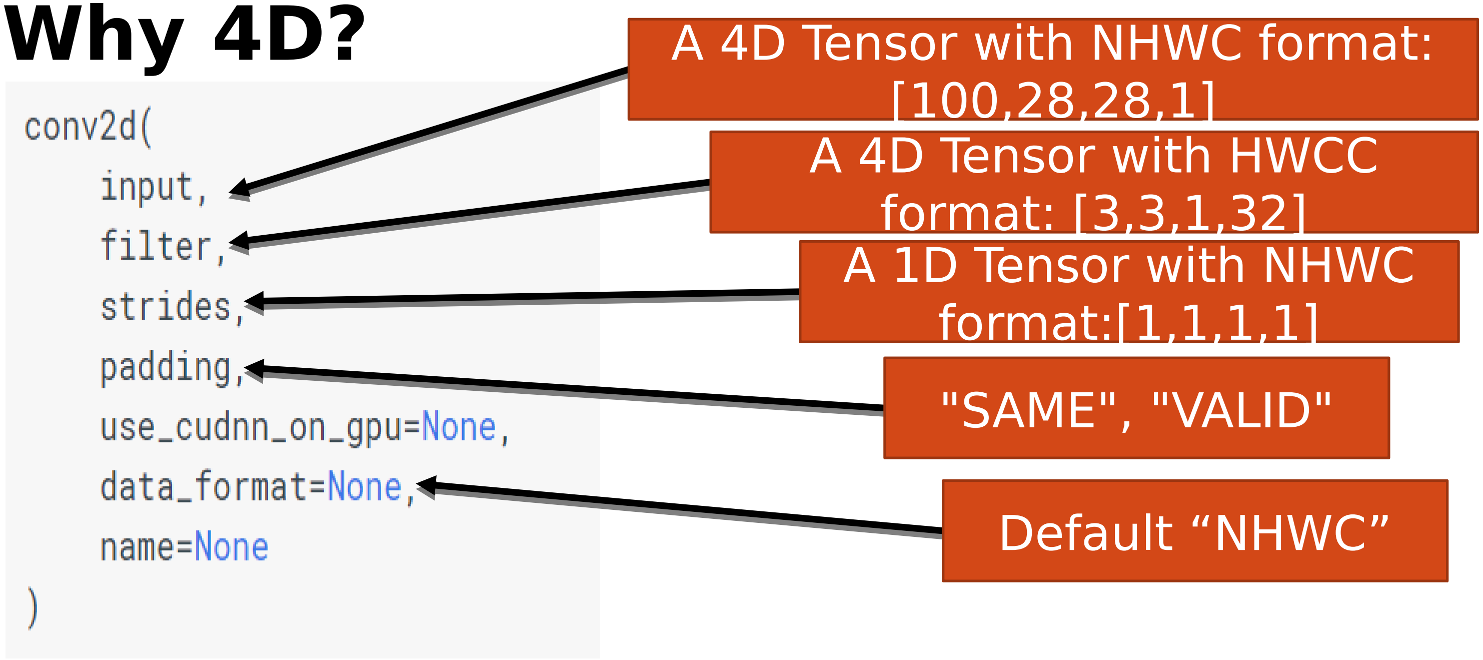
Test Your Knowledge



```
# Test trained model
correct_prediction = tf.equal(tf.argmax(y, 1), tf.argmax(y_, 1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
print(sess.run(accuracy, feed_dict={x: mnist.test.images,
                                    y_: mnist.test.labels}))
```

TensorFlow: conv2d

- **It computes a 2-D convolution given 4-D input and filter tensors**
- **Why 4D?**



```
conv2d(
    input,
    filter,
    strides,
    padding,
    use_cudnn_on_gpu=None,
    data_format=None,
    name=None
)
```

A 4D Tensor with NHWC format: [100,28,28,1]

A 4D Tensor with HWCC format: [3,3,1,32]

A 1D Tensor with NHWC format:[1,1,1,1]

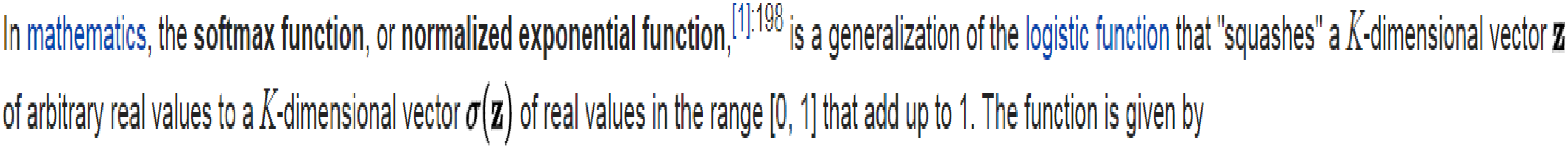
"SAME", "VALID"

Default “NHWC”

Demo

Backup

Softmax



In mathematics, the **softmax function**, or **normalized exponential function**,[1]:198 is a generalization of the logistic function that "squashes" a K-dimensional vector $\mathbf{z}$ of arbitrary real values to a K-dimensional vector $\sigma(\mathbf{z})$ of real values in the range [0, 1] that add up to 1. The function is given by

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1, \ldots, K.$$

"Euler's number“ (**2.71828)**

z = [1.0, 2.0, 3.0, 4.0, 1.0, 2.0, 3.0]

=e^{z_j}2.72, 7.39, 20.09, 54.6, 2.72, 7.39, 20.09]

$\sum_{k=1}^{K} e^{z_k}$ = 114.8

Add up to 1

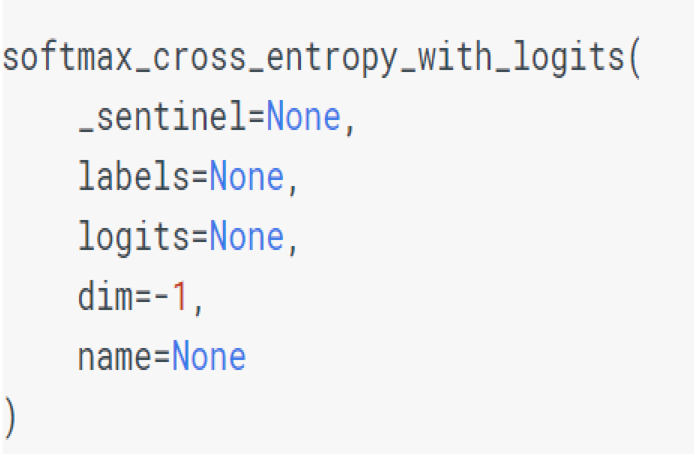
$\sigma(\mathbf{z})_j$ = [0.024, 0.064, 0.175, 0.475, 0.024, 0.064, 0.175]

How to use Reshape

- input_layer = tf.reshape(features["x"], [-1, 28, 28, 1])
- Note that we've indicated -1 for batch size, which specifies that this dimension should be dynamically computed based on the number of input values in features["x"], holding the size of all other dimensions constant. This allows us to treat batch_size as a hyperparameter that we can tune. For example, if we feed examples into our model in batches of 5, features["x"] will contain 3,920 values (one value for each pixel in each image), and input_layer will have a shape of [5, 28, 28, 1]. Similarly, if we feed examples in batches of 100, features["x"] will contain 78,400 values, and input_layer will have a shape of [100, 28 28, 1].

Cost Function

- tf.nn.softmax_cross_entropy_with_logits
- Format:



```
softmax_cross_entropy_with_logits(
    _sentinel=None,
    labels=None,
    logits=None,
    dim=-1,
    name=None
)
```

- Computes softmax cross entropy between logits and labels

The End