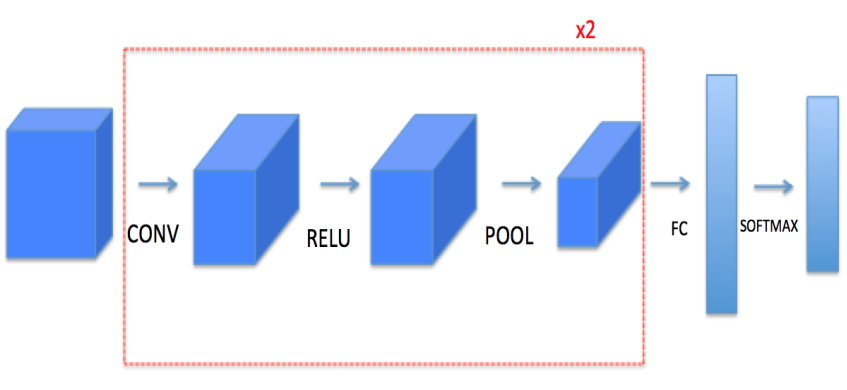
Convolutional neural network

# Convolutional model

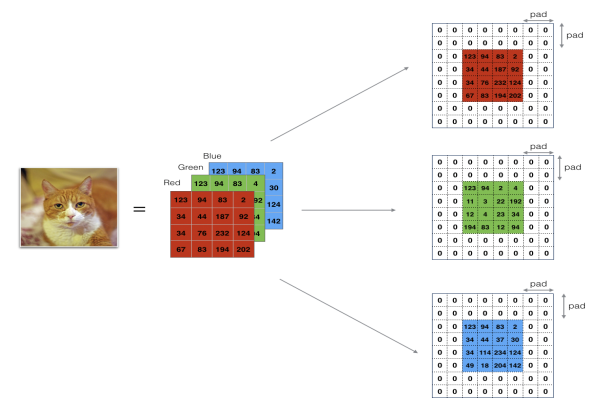


For every forward function, there is its corresponding backward equivalent. Hence, at every step of your forward module we will store some parameters in a cache. These parameters are used to computer gradients during backward propagation.

## convolve function

### zero padding

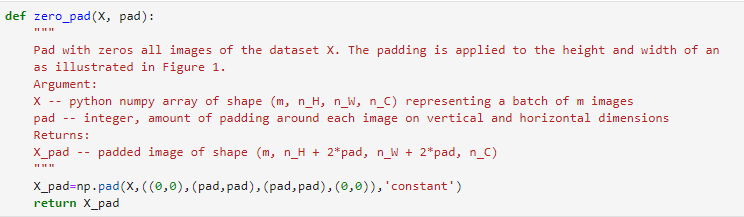
Zero-padding adds zeros around the broder of an image.



The main benefits of padding are the following: first, it allows you to use convolve layer without shrinking the height and width of the volume. That is same convolution. Secondly, it helps us keep more of edge information.

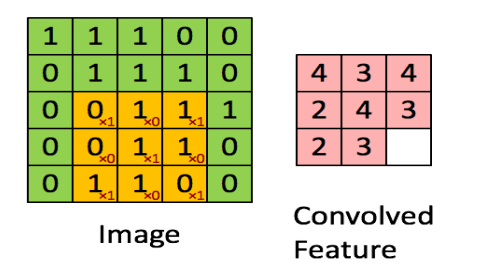
In zero pad () function, the input X is the python numpy array of shape (m, n\_H, n\_W, n\_C) representing a batch of m images.

The function is as following:

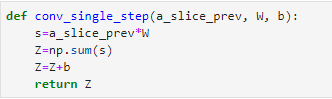


### convolve window

Convolve window at here is the f x f filter, which used in array to change the shape of the input image. In practice, we apply the filter to a single position of the input. This will be used to build a convolutional unit, which means a single step of convolution.

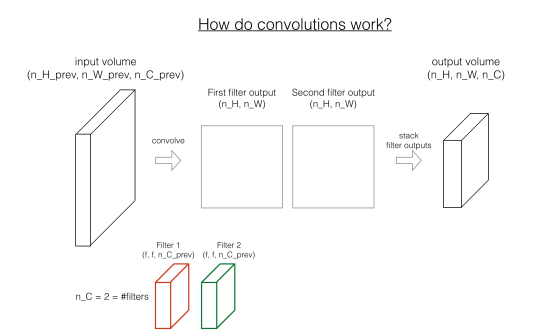


In this process, containing three steps:1) takes an input volume; 2) apply a filter at every position of the input; 3) output another volume. Pay attention the parameters in the function below. A previous slice should have the same shape if W.



### convolution forward

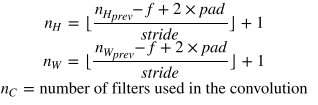
In the forward process, we will take several filters and convolve them with the input. A convolutional neural network works just like following:



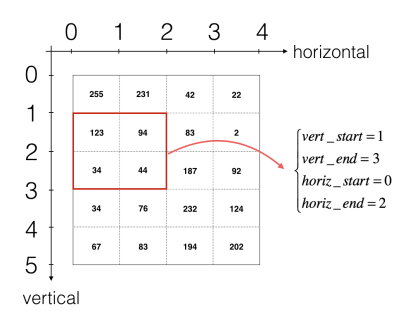
For every convolution, we classifier the parameters of input and output. In neural networks, W represents the filter. As to the shape of the W, channel should be the same with the input channel.

Giver a specific parameter, for example, a previous array contains height, width, class, and training number. W array contains filter number, previous class and the number of filters, which mean (f, f, previous c, c)

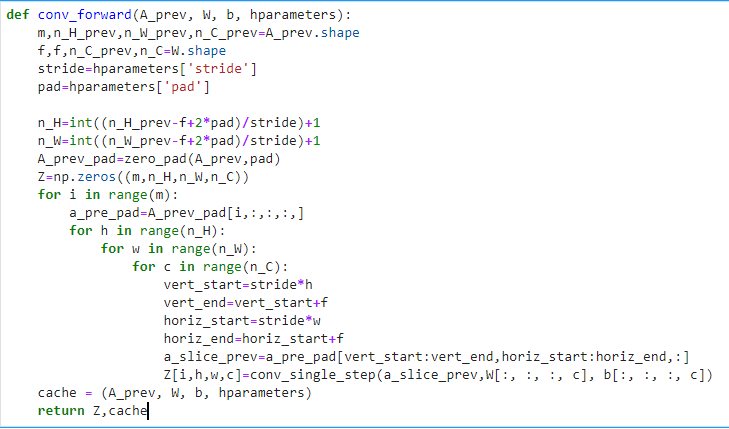
How to calculate the new shape after convolute process? We find out new height has germane with previous height, stride, f, and pad, we get the formula as following:



How to calculate a previous slice which has the same shape with W? We take an easy example into account.



We can see, in first time, the vertical axis starts with 0 and end with 0+f; horizontal axis starts 0 and end with 0+f; in the second time, the vertical axis starts with stride\*1 and end with stride\*1+f, the same with the horizontal axis.



### convolution backward

In convolution backward, we should calculate dA, dW, db. Before we do backward, let us look at convolution backward first.

As we all know, if we know array previous, weight and bias, we can calculate the new layer parameters.



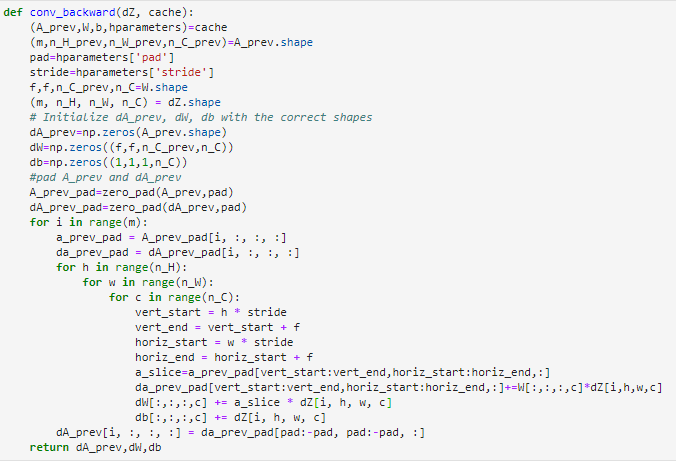
then we do derivative, we can get that:





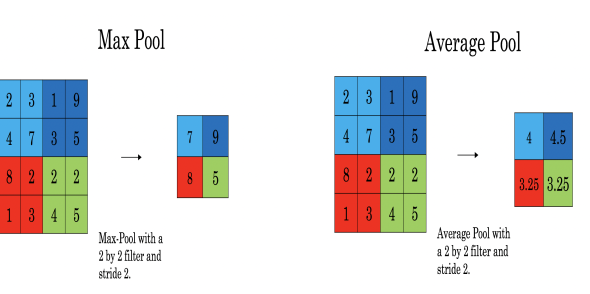






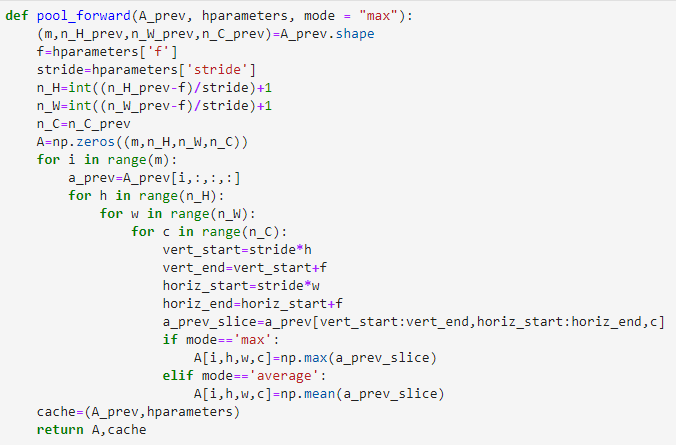
## pooling function

The function of a pooling layer is to reduce the height and width of the input image. There are two kinds of pooling functions, one is maximum pool and the other is average pool.



### Forward pooling

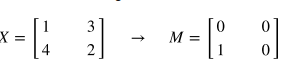
Forward pooling has the similar algorithm as convolve forward, the distinguish is that the former function do not change the channel. For example, the input image has the shape (m, h\_prev, w\_prev, c\_prev) and the shape of the filter is (f, f). After the pooling, the shape of output is (m, h, w, c\_prev). h and w has the same formula as convolve forward function.



### Backward pooling

Although in backward pooling, we have no parameter to update, but we get different array which may make difference of gradient.

Let us analysis the backward of maximum pooling and average pooling at first. In max operation, the function is as following:



We give this process a new name: create mask from window.



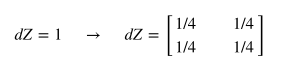
Then we can get the formula about A previous derivative as following



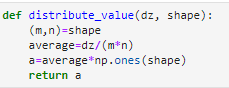


Then we look at average pooling.

At first, we see the simple example:



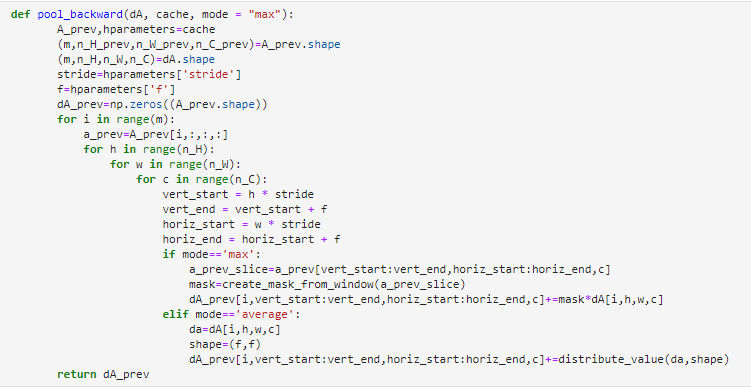
We call this process as distribution values, the purpose of this step is to return an array that fit for the previous array.



From the average pooling formula, we can calculate backward average pool.

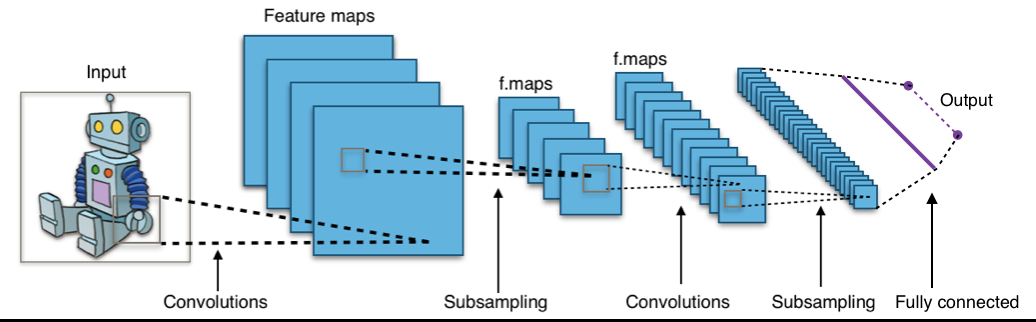


So the pool backward function is as following:

cte

# Convolutional application

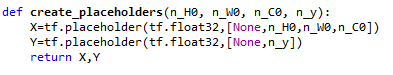
In previous course, we used soft max method to classifier a sign gesture. Since the input is the image, it is more natural to use convolution method to deal with such problems. At this task, we use deep learning framework tensor flow.



## Create placeholder

In tensor flow, we need to create placeholder for the input data that will be feed into the model whining running the session.

In convolution neural network, we know the input image has the shape of m, height, width and color channel, and the output image has the shape of m and a specific number, since the output should be a concrete result. So the definition of create placeholder is as following:

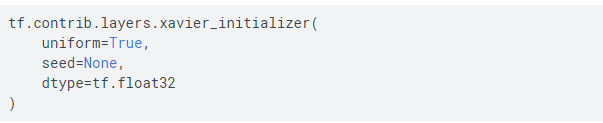


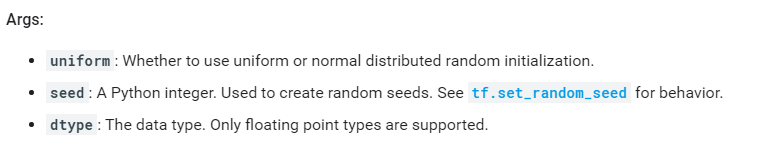
## Initialize parameters

In convolution neural network, we need to initialize W and b at first. The difference between the classifier and convolve is that W has D-dimensional and D is larger than 2.

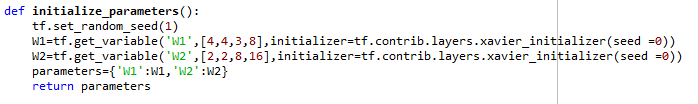
### Xavier initializer

This initializer is designed to keep the scale of the gradients roughly the same in all layers. The parameters contain uniform, seed and data type.





So the function of initialize parameters as follows:

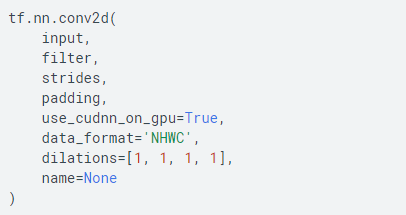


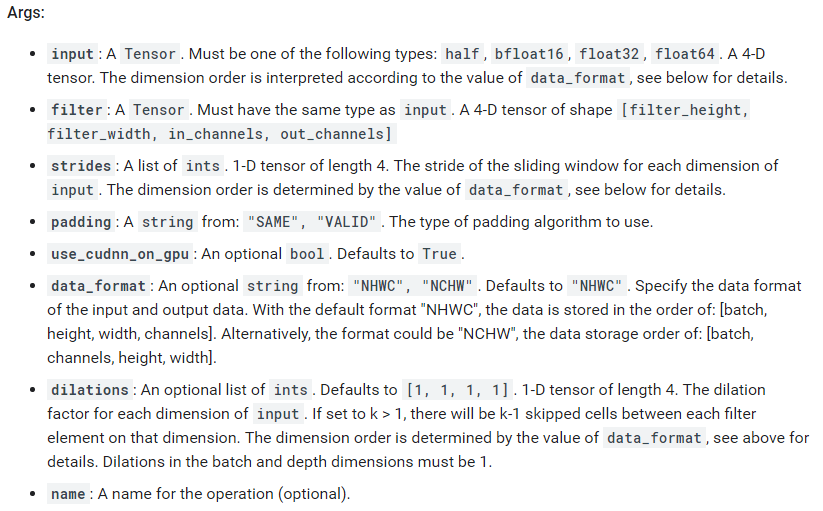
## Forward propagation

In forward propagation, we build the following model: CONV2D -> RELU -> MAXPOOL ->CONV2D -> RELU -> MAXPOOL -> FLATTEN -> FULLYCONNECTED.

### tf.nn.conv2d

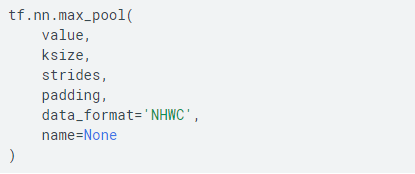
Give an input X, do convolve with filter W, and filter window move strides every time. Padding has two choices: same and valid. Same means make the output shape has the same the input image.

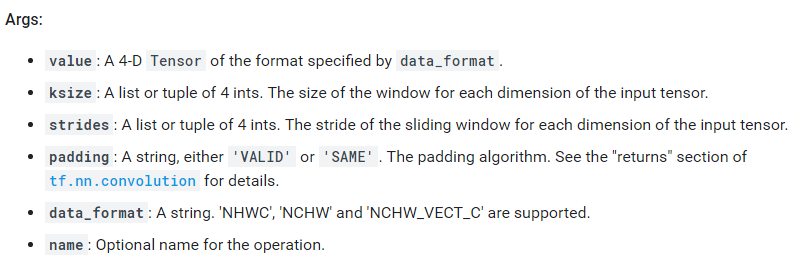




### tf.nn.max\_pool

Given an input A, this function uses a window of size (f, f) and strides of size (s, s) to carry out max pooling over each window.





### tf.nn.relu

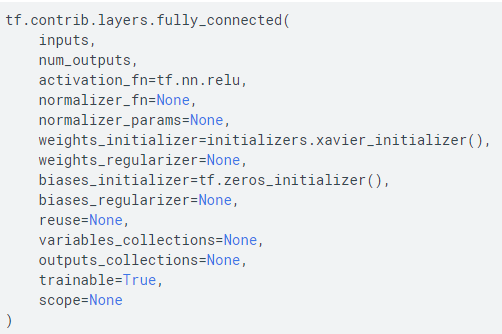
Computes the elementwise ReLU of Z1

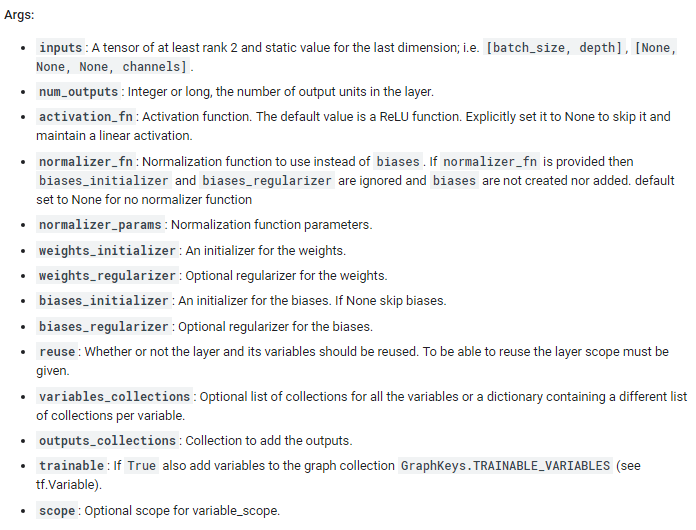
### tf.contrib.layers.flatten

Flatten the input while maintain the batches.

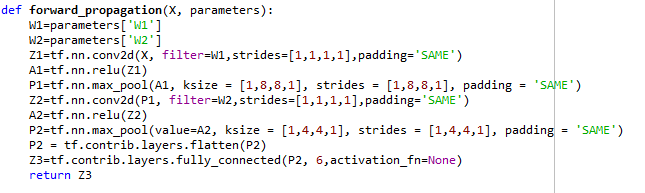
### tf.contrib.layers.fully\_connected

Fully connected creates a variable called weights, representing a fully connected weight matrix, which is multiplied by the inputs to produce a Tensor of hidden units. If a normalizer\_fn is provided (such as batch\_norm), it is then applied. Otherwise, if normalizer\_fn is None and a biases\_initializer is provided then a biases variable would be created and added the hidden units. Finally, if activation\_fn is not None, it is applied to the hidden units as well.





So the function of forward propagation is as follows:



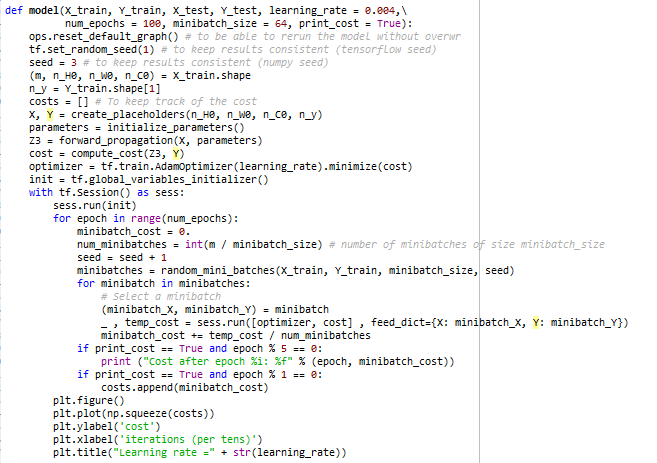
## Cost function

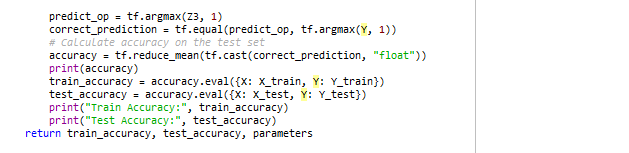


## Model

In model process, we usually take the following steps:

1. Create placeholder: deal with input and output parameters
2. Initialize parameters: deal with W and b
3. Forward propagation
4. Compute cost
5. Choose optimizer
6. Set global initialize variable
7. Run in tensor flow session: for loop, and random mini batch to get X and Y, using operation system to get lower cost.
8. Predict the accuracy of training and test data





## Result analysis

Train Accuracy: 0.9787037

Test Accuracy: 0.85

