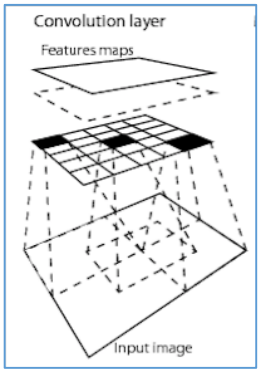
TensorFlow Report
Name: Rodrigo Joni Sestari
Matricola: 179020

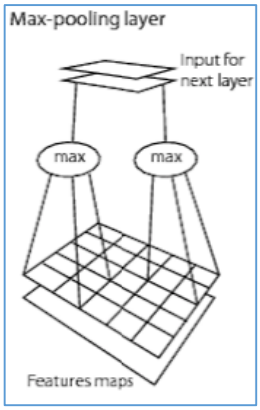
1. **Methodology**:

- **Convolution Layer**:



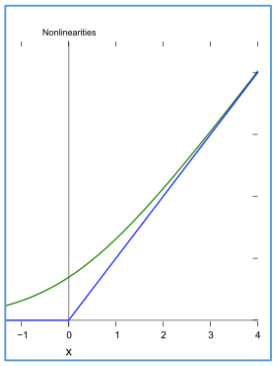
Consist of a rectangular grid of neurons. It requires that the previous layer also be a rectangular grid of neurons. Each neuron takes inputs from a rectangular section of the previous layer, the weights for this rectangular section are the same for each neuron in the convolutional layer.

- **Max Pooling**:



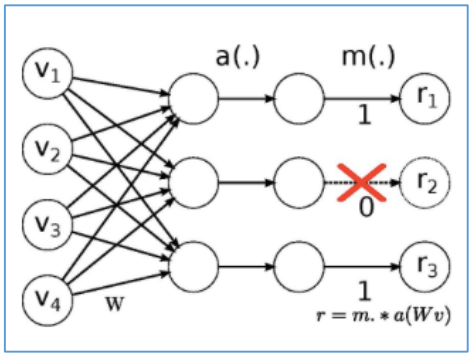
After each convolutional layer, there may be a pooling layer. The pooling layer takes small rectangular blocks from the convolutional layer and subsamples it to produce a single output from that block. There are several ways to do this pooling, such as taking the average or the maximum, or a learned linear combination of the neurons in the block.

- **ReLU**:



Rectified Linear Units, it is an activation function defined as f(x) =max(0,x), it come used as sigmoid function. the major different between both is that sigmoid has a rang [0,1] while ReLU [0,inf].

- **Dropout**



Deep neural networks with a large number of parameters are very powerful machine learning systems. However, overfitting is a serious problem in such networks. Large networks are also slow to use, making it difficult to deal with overfitting by combining the predictions of many different large neural nets at test time. Dropout is a technique for addressing this problem. The key idea is to randomly drop units (along with their connections) from the neural network during training.

- **Softmax**:

  Softmax is also known as multinomial logistic regression. This model generalizes logistic regression to classification problems where the class label *y* can take on more than two possible values. The prediction for y_{pred} can be write who:

$$y_{pred} = \text{argmax}_i P(Y = i|x, W, b)$$

2. **Work**

- **Model 1**: I simply deleted the first layer and then I changed the input value of second layer from 32 to 1:

```
#2st layer
W_conv2 = weight_variable([5, 5, 1, 64])
b_conv2 = bias_variable([64])

h_conv2 = tf.nn.relu(conv2d(x_image, W_conv2) + b_conv2)
h_pool2 = max_pool_2x2(h_conv2)
```

- **Model 2:** How in the first model, I changed the value of input of layer 3 from 7x7 to 14x14. This model used MaxPoling only one time, then the image was transformed from 28x28 to 14x14.

```
#3st layer
W_fc1 = weight_variable([14 * 14 * 64, 1024])
b_fc1 = bias_variable([1024])

h_pool2_flat = tf.reshape(h_pool2, [-1, 14 * 14 * 64])
h_fc1 = tf.nn.relu(tf.matmul(h_pool2_flat, W_fc1) + b_fc1)
```

- **Model 3**: It was a bit complicated, because I can't use directly the output of layer 2 in the layer 3 because is necessary reshape the input as a flat vector, so a leave the line of reshape and changed the input of layer 4.

```
#3st layer
# W_fc1 = weight_variable([7 * 7 * 64, 1024])
# b_fc1 = bias_variable([1024])

h_pool2_flat = tf.reshape(h_pool2, [-1, 7*7*64])
# h_fc1 = tf.nn.relu(tf.matmul(h_pool2_flat, W_fc1) + b_fc1)

#add dropout
# keep_prob = tf.placeholder("float")
# h_fc1_drop = tf.nn.dropout(h_fc1, keep_prob)



#4st layes
W_fc2 = weight_variable([7*7*64, 10])
b_fc2 = bias_variable([10])

y_hat = tf.nn.softmax(tf.matmul(h_pool2_flat, W_fc2) + b_fc2)
```

I also comment the lines bellow, because dropout was used in the accuracy:

```
    train_accuracy = accuracy.eval(feed_dict={x:batch[0], y: batch[1] }) #, keep_prob: 1.0})
    print("step %d, training accuracy %g" % (n, train_accuracy))
  sess.run(train_step,feed_dict={x: batch[0], y: batch[1] })#, keep_prob: 0.5})

print("test accuracy %g" % accuracy.eval(feed_dict={ x: mnist.test.images, y: mnist.test.labels }))# ,keep_prob: 1.0}))
```

3. Result

Model	1	2	3
Accuracy	0.9895	0.9899	0.9878