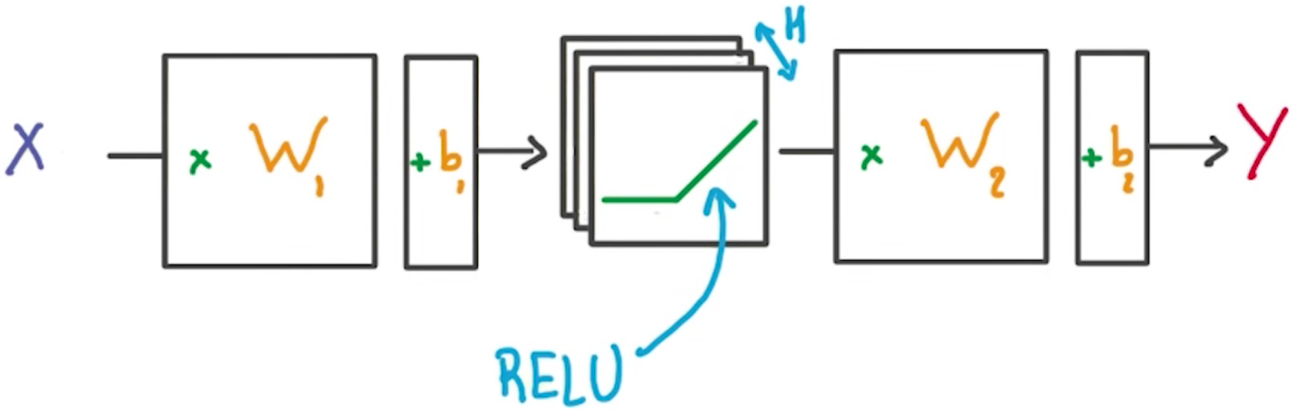
vectors of 784px.

Multilayer Perceptron




```
# Hidden Layer with RELU activation
layer_1 = tf.add(tf.matmul(x_flat, weights['hidden_layer']),\
    biases['hidden_layer'])
layer_1 = tf.nn.relu(layer_1)
# Output Layer with linear activation
logits = tf.add(tf.matmul(layer_1, weights['out']), biases['out'])
```

You've seen the linear function `tf.add(tf.matmul(x_flat, weights['hidden_layer']), biases['hidden_layer'])` before, also known as `xw + b`. Combining linear functions together using a ReLU will give you a two layer network.

Optimizer

```
# Define Loss and optimizer
cost = tf.reduce_mean(\
    tf.nn.softmax_cross_entropy_with_logits(logits=logits, labels=y))
optimizer = tf.train.GradientDescentOptimizer(learning_rate=learning_rate)\
    .minimize(cost)
```

This is the same optimization technique used in the Intro to TensorFLow lab.

Session