1. 因為 sigmoid 的 output 範圍是(0,1)，在 backpropagation 的時候，梯度容易因為太多層使得其趨近於 0，導致梯度消失，而 ReLU 的梯度都是 1 或 0，較容易可以減少這種風險。

2. $$floor\left(\frac{W + 2 \times pad - ks}{S}\right) + 1$$

floor(1.1)=1, floor(1.6)=1

```
convolution_param {
 num_output: 32
 pad: 1
 kernel_size: 3
 stride: 2
 }
```

```
# build the model
model = Model()
model.add(Conv(filter_size=3, input_channel=1, output_channel=8, pad=1, stride=2))  # Input has 1 channel
model.add(Activation("relu", None))
model.add(MaxPool(pool_size=2, stride=2))
model.add(Flatten())
model.add(Dense(8 * 8 * 8, 1))  # Adjust input_dim from Flatten
model.add(Activation("sigmoid", None))
```

因為我不擅長通靈，所以試幾次後，最後我決定參考了網路上的方法。

在選值的過程中，我發現如果 pool_size 選很小，跑超慢，可能是因為減少的 feature 就會比較少所以變超久；filter_size 則是越大跑越慢，我猜測是因為計算量的問題；然後 layer 太多，learning 太小，速度直接慢到起床了還沒 train 好。

3.

```
loss_function = 'cross_entropy'
layers_dims = [784, 128, 64, 32, 1]
activation_fn = ['relu', 'relu', 'relu', 'sigmoid']
learning_rate = 0.01
num_iterations = 1000
print_loss = True
print_freq = 200
decrease_freq = 2000
decrease_proportion = 0.8
# You might need to use mini_batch to reduce training time
batch_size = 32
```

NN model parameters -> 784*128 + 128*64 + 64*32 + 32*1 = 110624

CNN model parameters -> 3*3*1*8 + 8*8*8*1 = 584

Compare -> 110624 – 584 = 110040 ????