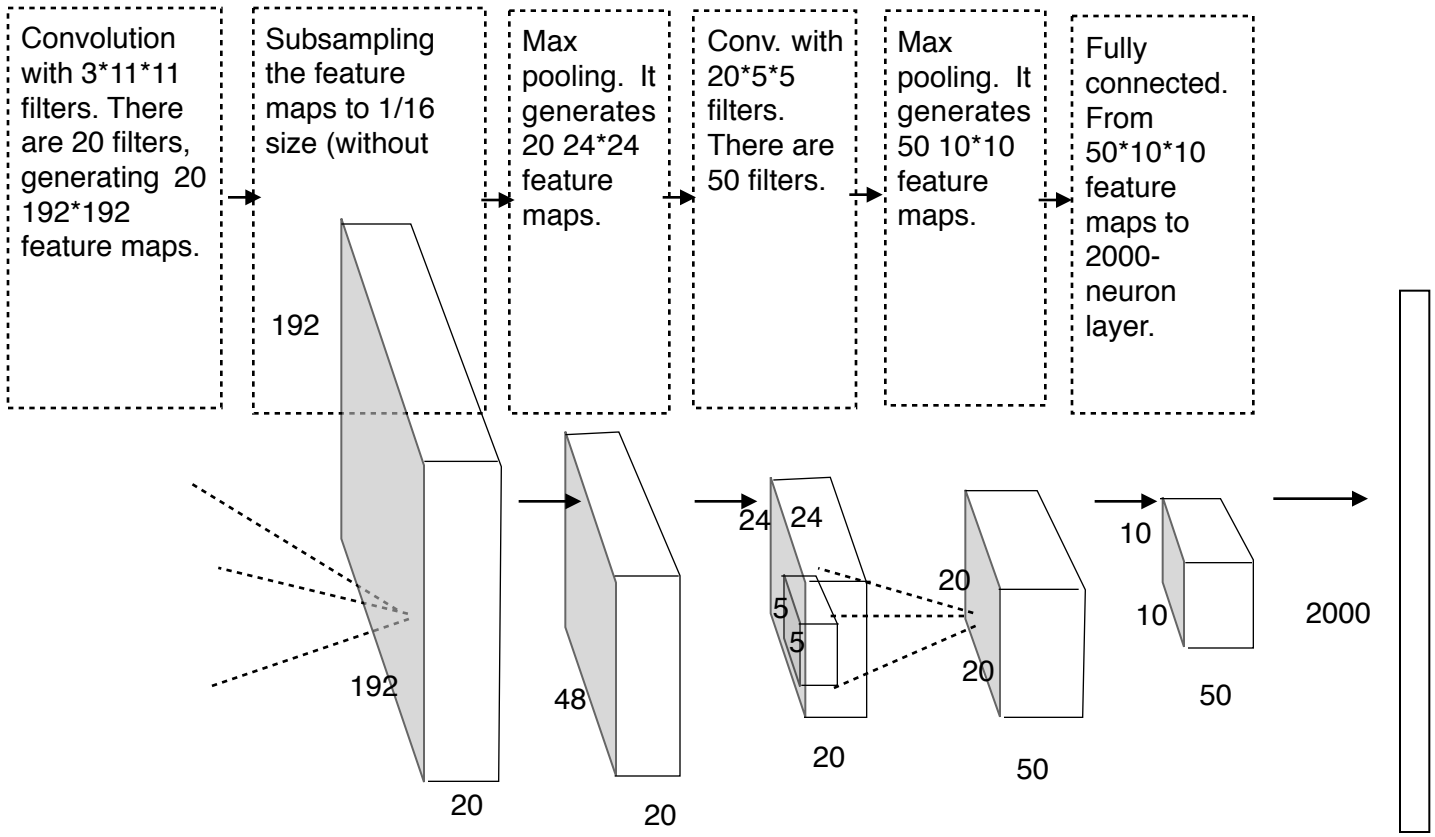
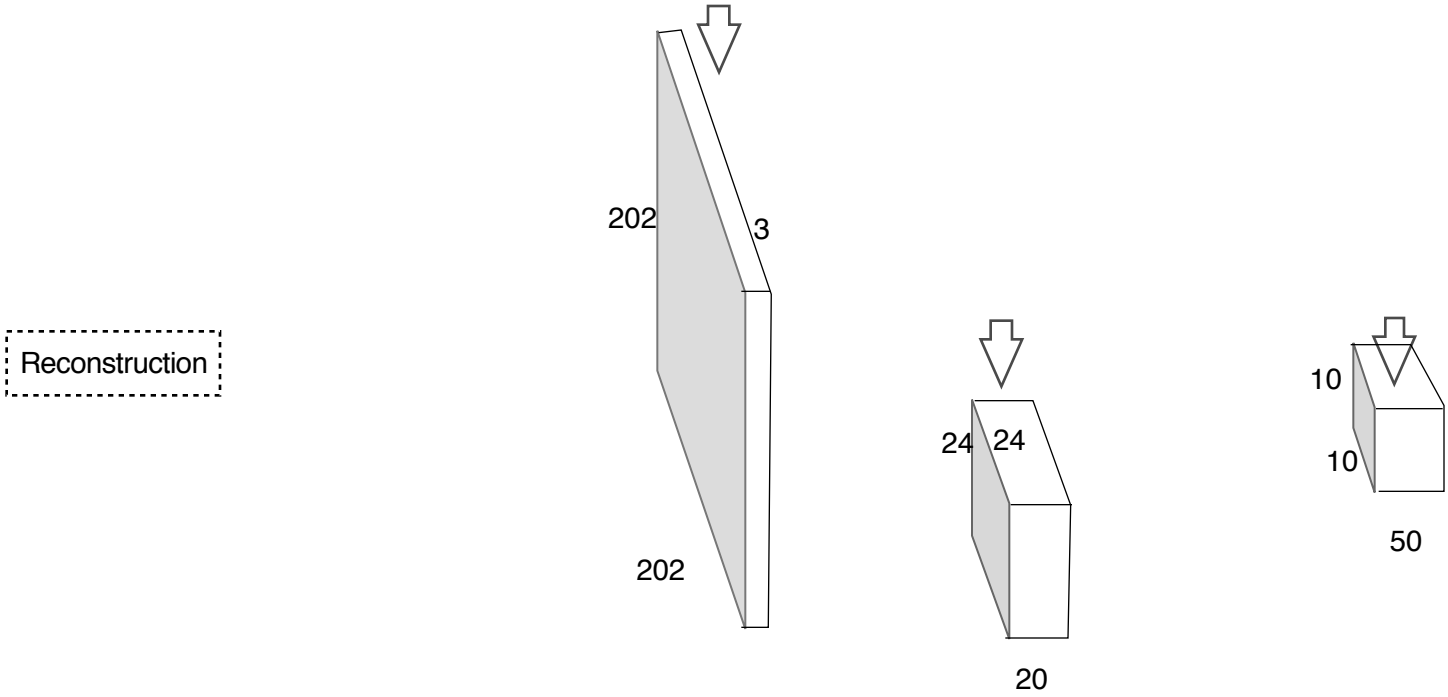
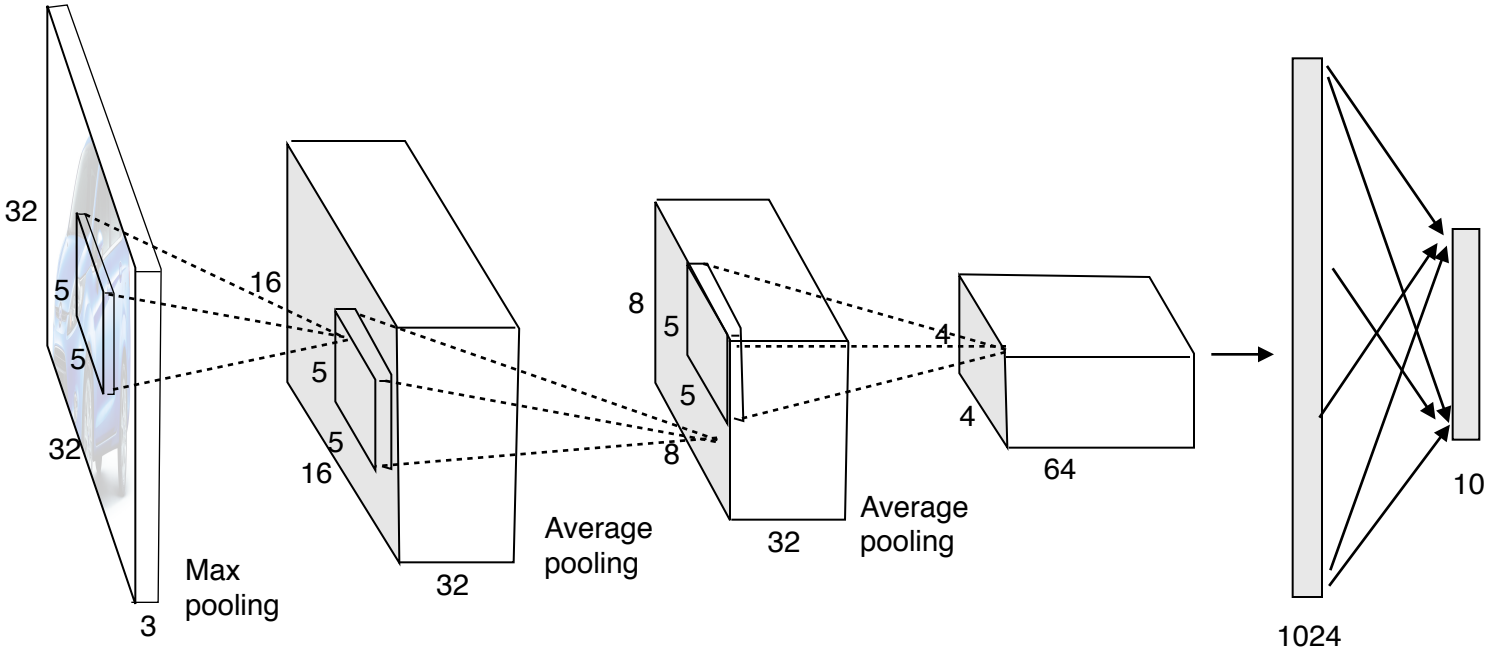
1. Pre-training with Auto-encoder

The structure of the network is like the picture below **(I think there're some mistake of this design, see Section 4)** :







Three tied-weight auto-encoders are stacked.

The first two are convolutional auto-encoders. For a mono-channel input x, k-th feature map is:

$$h_k=f(x*W_k+b_k)$$

where *f* is activation function **tanh**. Convolution is a valid convolution. The reconstruction is:

$$y=f(\sum_{k\in H} h_k*W_{k^{\sim}}+c_k)$$

where $W_{k^{\sim}}$ is the flip of W_k over both dimensions, and convolution is full convolution. This way the weights are tied.

The last auto-encoder is basic one.

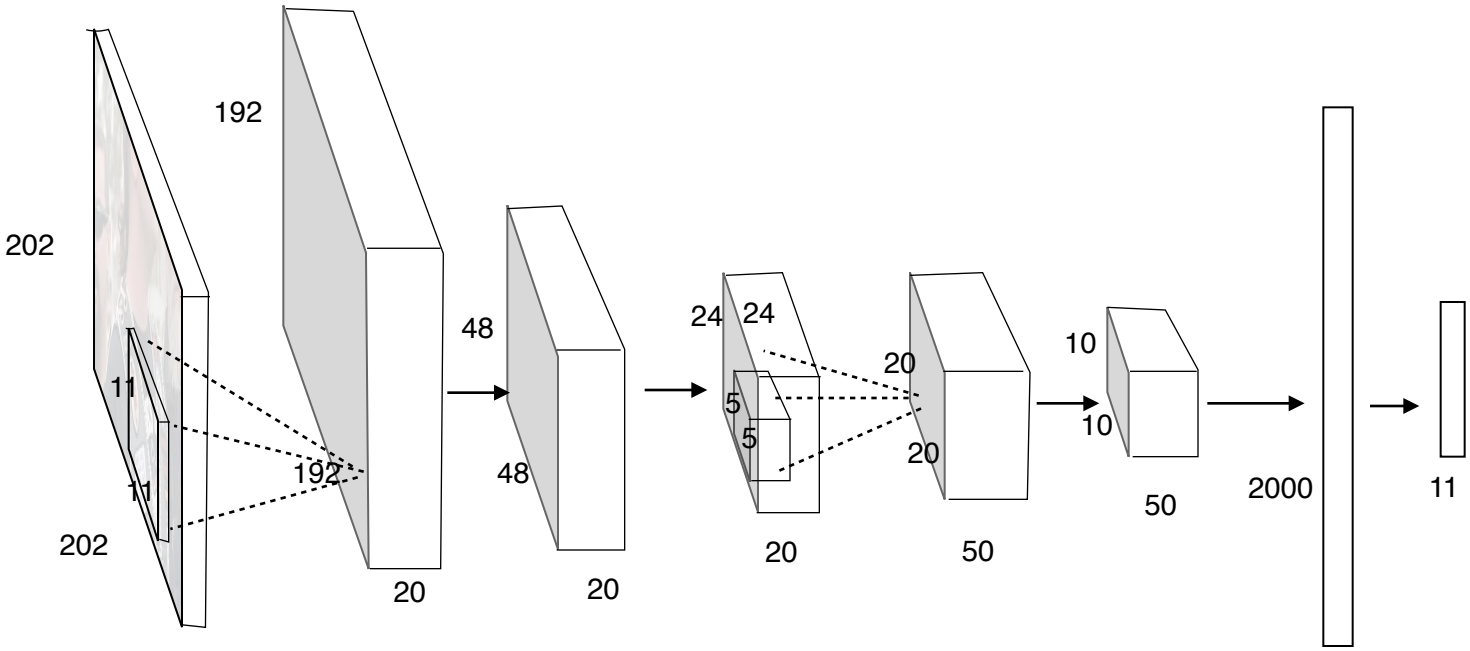
The loss function is mean squared error:

$$E=\sum_{i=1}^{n}(x_i-y_i)^2$$

I train the auto-encoder in layer-wise fashion: First layer, and then second layer, finally last layer. Stochastic gradient descent is used. A mini-batch of training data is input at one time. In the convolutional auto-encoder, parameters are (W, b, c), and gradient of the loss function with respect to them is calculated.

2. Training the Conv-net

The structure of the convolutional network is similar with the stacked auto-encoder, just without the reconstruction and with an extra logistic regression layer in the end. It's as picture below:



The parameters obtained in the pre-training process is used here as initial parameter values. The parameters (*W,b*) in the logistic regression layer is set to zero. Still, stochastic gradient descent with mini-batches is used. Loss function is the negative log likelihood.

3. What I'm doing

I made some mistake when creating the dataset, so the previous results should not be right. I corrected the mistake, and hopefully will come to some new results before our next meeting. Also, I'll expand the dataset to 1035 scene images from 849 images. This is because the original dataset was downloaded when I was in China, and many links cannot be accessed in China, now I re-download the dataset and can expand the images to 1035. This is not a huge expansion, so I don't think it will change the result much.

I think what I'm doing wrong in the current network is that I'm not actually adding max-pooling into the auto-encoder. It doesn't participate in the process of reconstruction now. This is because when I designed the network, I just wanted to build an easier one and test whether it works, and if it works, then move to more complicated one taken max-pooling into consideration. However, I now think that max-pooling may be the most important step and shouldn't be eliminated. (Max-pooling is where dimension reduction is performed, and what auto-encoder should do is to reconstruct the input from output after max-pooling)

I'll also use denoising auto-encoder instead of the basic one.