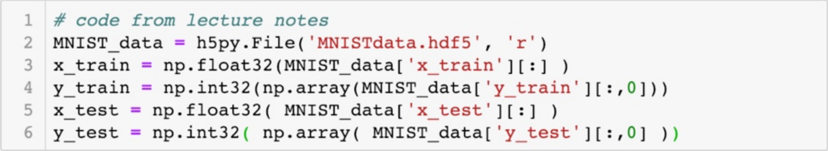
CS398-Deep Learning
Homework 3
Lingyi Xu (lingyix2)

1. Description of Implementation

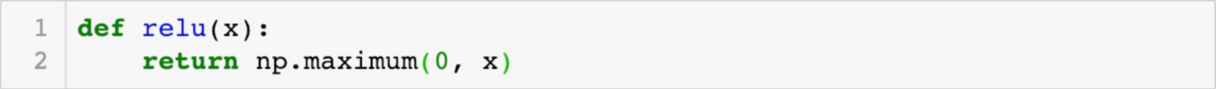
First, I load the MNIST data from the dataset.



```python
# code from lecture notes
MNIST_data = h5py.File('MNISTdata.hdf5', 'r')
x_train = np.float32(MNIST_data['x_train'][:] )
y_train = np.int32(np.array(MNIST_data['y_train'][:,0]))
x_test = np.float32( MNIST_data['x_test'][:] )
y_test = np.int32( np.array( MNIST_data['y_test'][:,0] ))
```

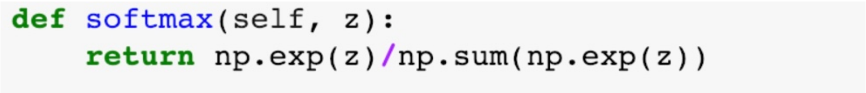
Then I build my mini-batch CNN model.

I define the nonlinearities σ(z) (relu) function I will use:



```python
def relu(x):
    return np.maximum(0, x)
```

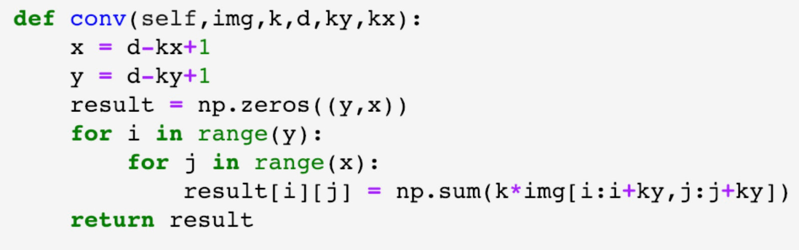
The softmax function I will use:



```python
def softmax(self, z):
    return np.exp(z)/np.sum(np.exp(z))
```

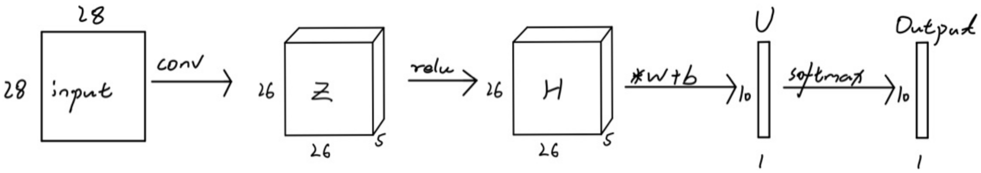
The convolution layer:

$$(X * K)_{i,j} = \sum_{m=0}^{k_y-1} \sum_{n=0}^{k_x-1} K_{m,n} X_{i+m,j+n}.$$



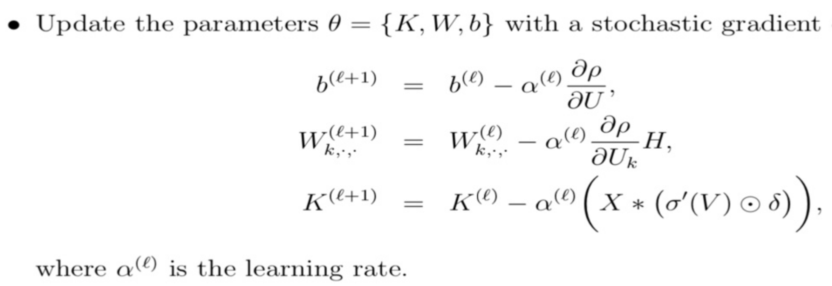
```python
def conv(self,img,k,d,ky,kx):
    x = d-kx+1
    y = d-ky+1
    result = np.zeros((y,x))
    for i in range(y):
        for j in range(x):
            result[i][j] = np.sum(k*img[i:i+ky,j:j+ky])
    return result
```

My one-layer CNN model looks like this:



Accordingly, I defined the forward propagation step.

For the backward propagation step, according to the lecture note:



- Update the parameters $\theta = \{K, W, b\}$ with a stochastic gradient descent step:

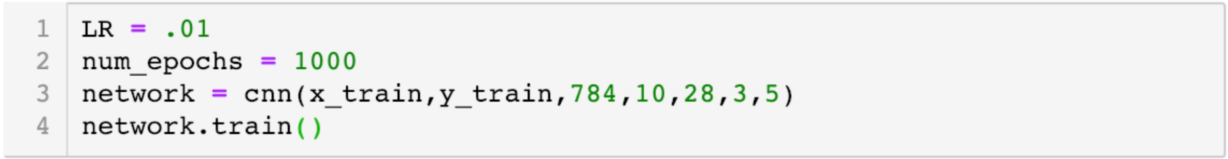
$$
\begin{aligned}
b^{(\ell+1)} &= b^{(\ell)} - \alpha^{(\ell)} \frac{\partial \rho}{\partial U}, \\
W^{(\ell+1)}_{k,\cdot,\cdot} &= W^{(\ell)}_{k,\cdot,\cdot} - \alpha^{(\ell)} \frac{\partial \rho}{\partial U_k} H, \\
K^{(\ell+1)} &= K^{(\ell)} - \alpha^{(\ell)} \Big(X * \left(\sigma'(V) \odot \delta \right) \Big),
\end{aligned}
$$

where $\alpha^{(\ell)}$ is the learning rate.

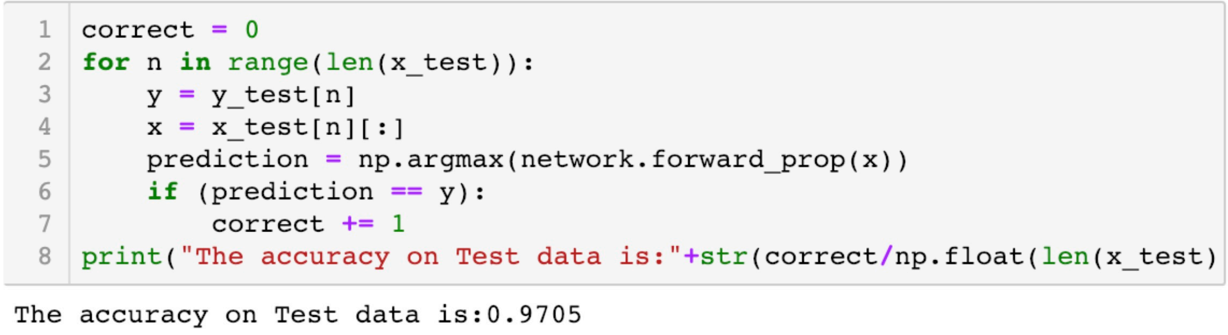
For the forward and backward propagation, see “HW3.ipynb”.
For this particular dataset, I use the following parameters:
Iteration = 1000
Batch size = 150
Number of channel = 5
Learning rate = 0.01/（0.01*itr+1）

2. Final Test Accuracy:

97.05%



```
LR = .01
num_epochs = 1000
network = cnn(x_train,y_train,784,10,28,3,5)
network.train()
```



```
correct = 0
for n in range(len(x_test)):
    y = y_test[n]
    x = x_test[n][:]
    prediction = np.argmax(network.forward_prop(x))
    if (prediction == y):
        correct += 1
print("The accuracy on Test data is:"+str(correct/np.float(len(x_test))
```

```
The accuracy on Test data is:0.9705
```