CS 598 Homework 2

Mohan Sun[*]

We implement a neural network with a single convolution layer with 5 filters, and train the model with MNIST data. We achieved an accuracy of 95.13% which satisfies the requirement. Figure 1 shows training results and timing.

Model the model is a modified and extended version of Logisticregression.py. We keep the i/o and data processing part, keep the structure of Epoch loop, mini-batch loop, and forward as well as backward as function. We modified forward to:

```
def forward(x,y, model):
    Z = convolution(x, model['K'])
    H = sigmoid(Z)
    U = np.einsum('hijk,ijk',model['W'],H).reshape( \
        model['W'].shape[0],1) + model['b']
    p = softmax_function(U)
    return Z, H, p
```

[*]As usual, the article uses "we" and "our", but the homework is done by only Mohan "Fred" Sun without any collaboration. Netid: mohans2

and backward to:

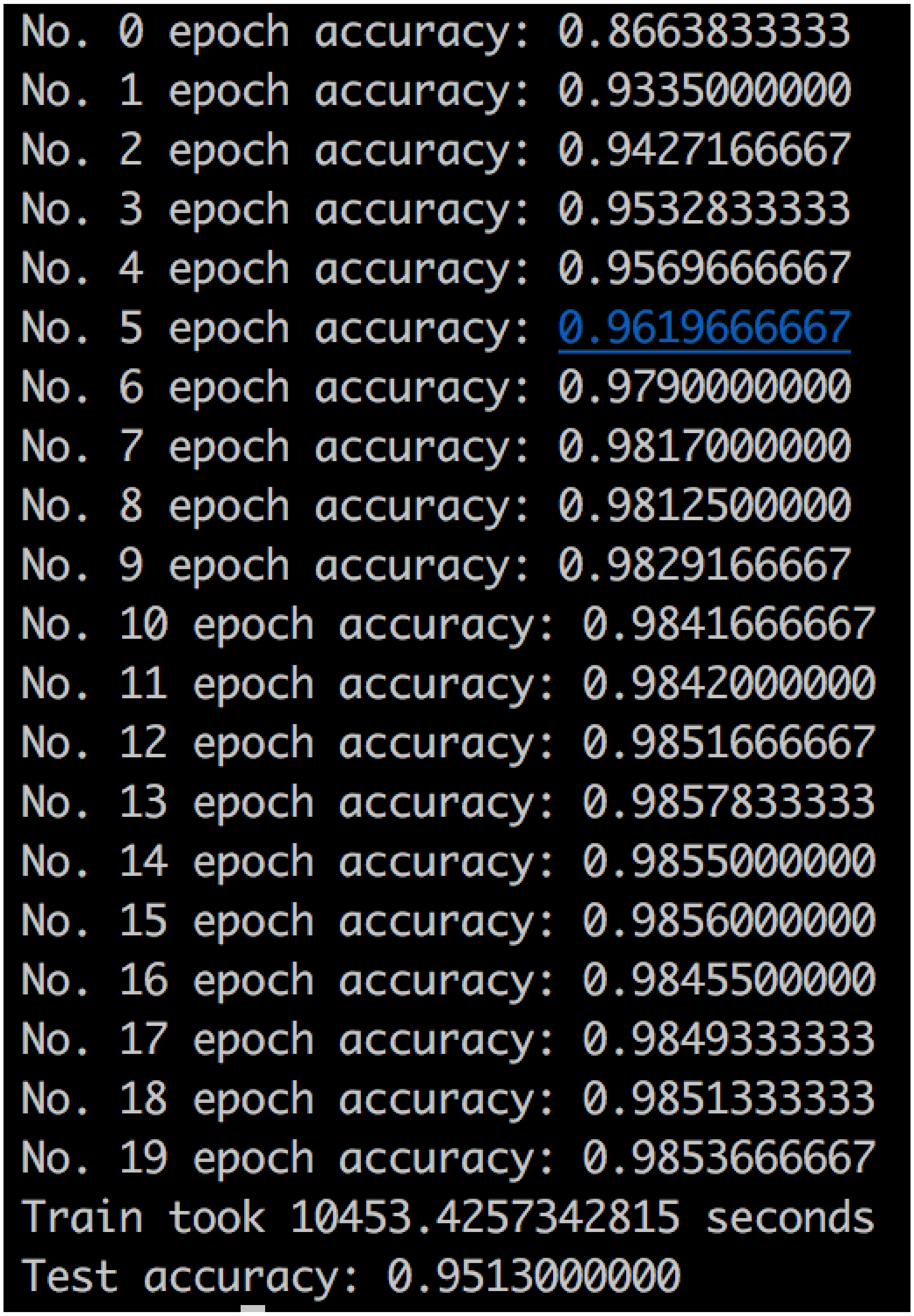
```
def backward(x, y, Z, H, p, model, model_grads):
    dU = np.copy(p)
    dU[y] = dU[y] - 1
    delta = np.einsum('hijk,h',model['W'], \
        dU.reshape(dU.shape[0]))
    dK = convolution(x, delta * sigmoidp(Z))
    dW = np.zeros_like(model['W'])
    for i in range(dW.shape[0]):
        dW[i,:,:,:] = dU[i] * H
    model_grads['W']=dW
    model_grads['b']=dU
    model_grads['K']=dK
    return model_grads
```

we also add a convolution function:

```
def convolution(X, K):
    dx, dy = X.shape
    kx, ky, p = K.shape
    ans = np.zeros((dx-kx+1,dy-ky+1,p))
    for k in range(ans.shape[2]):
        for j in range(ans.shape[1]):
            for i in range(ans.shape[0]):
                ans[i,j,k] = np.einsum('ij,ij', \
                    X[i:i+kx,j:j+ky], K[:,:,k])
    return ans
```

We still use sigmoid as ReLU tend to go overflow in softmax function.

parameters For this model, we define $\mathbf{K} \in \mathbb{R}^{k_x \times k_y \times C}$, $k_x = k_y = 5, C = 5$ and $LR^{(0)} = 0.1$. LR is scaled down by 10 for every 5 epochs.



```
No. 0 epoch accuracy: 0.8663833333
No. 1 epoch accuracy: 0.9335000000
No. 2 epoch accuracy: 0.9427166667
No. 3 epoch accuracy: 0.9532833333
No. 4 epoch accuracy: 0.9569666667
No. 5 epoch accuracy: 0.9619666667
No. 6 epoch accuracy: 0.9790000000
No. 7 epoch accuracy: 0.9817000000
No. 8 epoch accuracy: 0.9812500000
No. 9 epoch accuracy: 0.9829166667
No. 10 epoch accuracy: 0.9841666667
No. 11 epoch accuracy: 0.9842000000
No. 12 epoch accuracy: 0.9851666667
No. 13 epoch accuracy: 0.9857833333
No. 14 epoch accuracy: 0.9855000000
No. 15 epoch accuracy: 0.9856000000
No. 16 epoch accuracy: 0.9845500000
No. 17 epoch accuracy: 0.9849333333
No. 18 epoch accuracy: 0.9851333333
No. 19 epoch accuracy: 0.9853666667
Train took 10453.4257342815 seconds
Test accuracy: 0.9513000000
```

Figure 1: **Neural Network Training Result and Timing.** Training takes approximately 10453 seconds. We achieved an accuracy of 95.13%