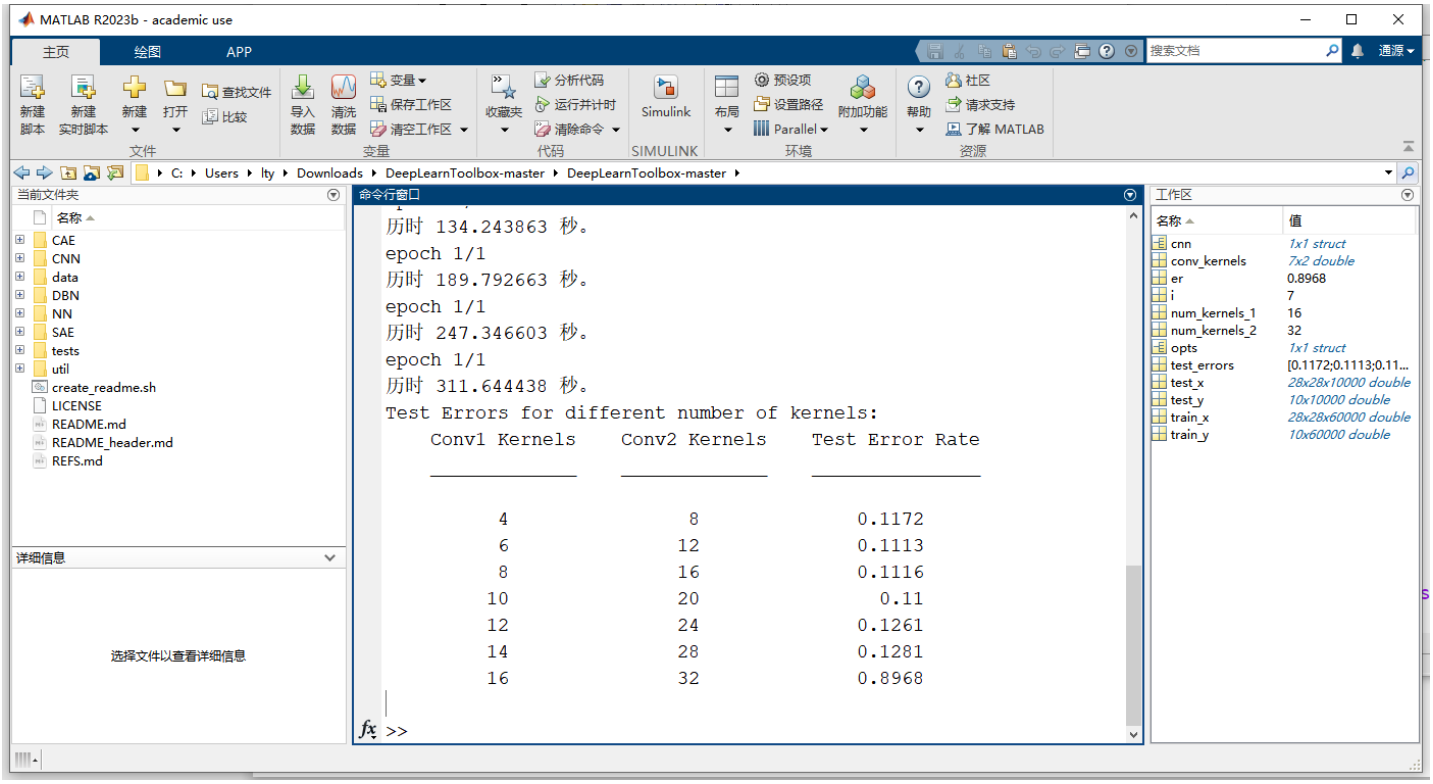
1

代码：

```
load mnist_uint8;

train_x = double(reshape(train_x', 28, 28, 60000)) / 255;
test_x = double(reshape(test_x', 28, 28, 10000)) / 255;
train_y = double(train_y');
test_y = double(test_y');

opts.alpha = 1;
opts.batchsize = 50;
opts.numepochs = 1;

% 定义不同的卷积核个数配置，使用元组（第一个卷积层卷积核个数，第二个卷积层卷积核个数）
conv_kernels = [4, 8; 6, 12; 8, 16; 10, 20; 12, 24; 14, 28; 16, 32];
test_errors = zeros(size(conv_kernels, 1), 1); % 存储每种配置下的测试错误率

for i = 1:size(conv_kernels, 1)
    num_kernels_1 = conv_kernels(i, 1);
    num_kernels_2 = conv_kernels(i, 2);

    rand('state', 0)
    cnn.layers = {
        struct('type', 'i') % 输入层
        struct('type', 'c', 'outputmaps', num_kernels_1, 'kernelsize', 5) % 第一个卷积层
        struct('type', 's', 'scale', 2) % 第一个子采样层
        struct('type', 'c', 'outputmaps', num_kernels_2, 'kernelsize', 5) % 第二个卷积层
        struct('type', 's', 'scale', 2) % 第二个子采样层
    };
    cnn = cnnsetup(cnn, train_x, train_y);

    cnn = cnntrain(cnn, train_x, train_y, opts);

    [er, ~] = cnntest(cnn, test_x, test_y);
    test_errors(i) = er;
end

% 绘制卷积核个数与测试错误率的关系图
figure;
plot(conv_kernels(:, 1), test_errors, '-o');
xlabel('Number of Kernels in First Conv Layer');
ylabel('Test Error Rate');
title('Test Error Rate vs Number of Kernels in First Conv Layer');
grid on;

```

```
disp('Test Errors for different number of kernels:');
disp(table(conv_kernels(:, 1), conv_kernels(:, 2), test_errors,
'VariableNames', {'Conv1 Kernels', 'Conv2 Kernels', 'Test Error Rate'}));
```

结果：



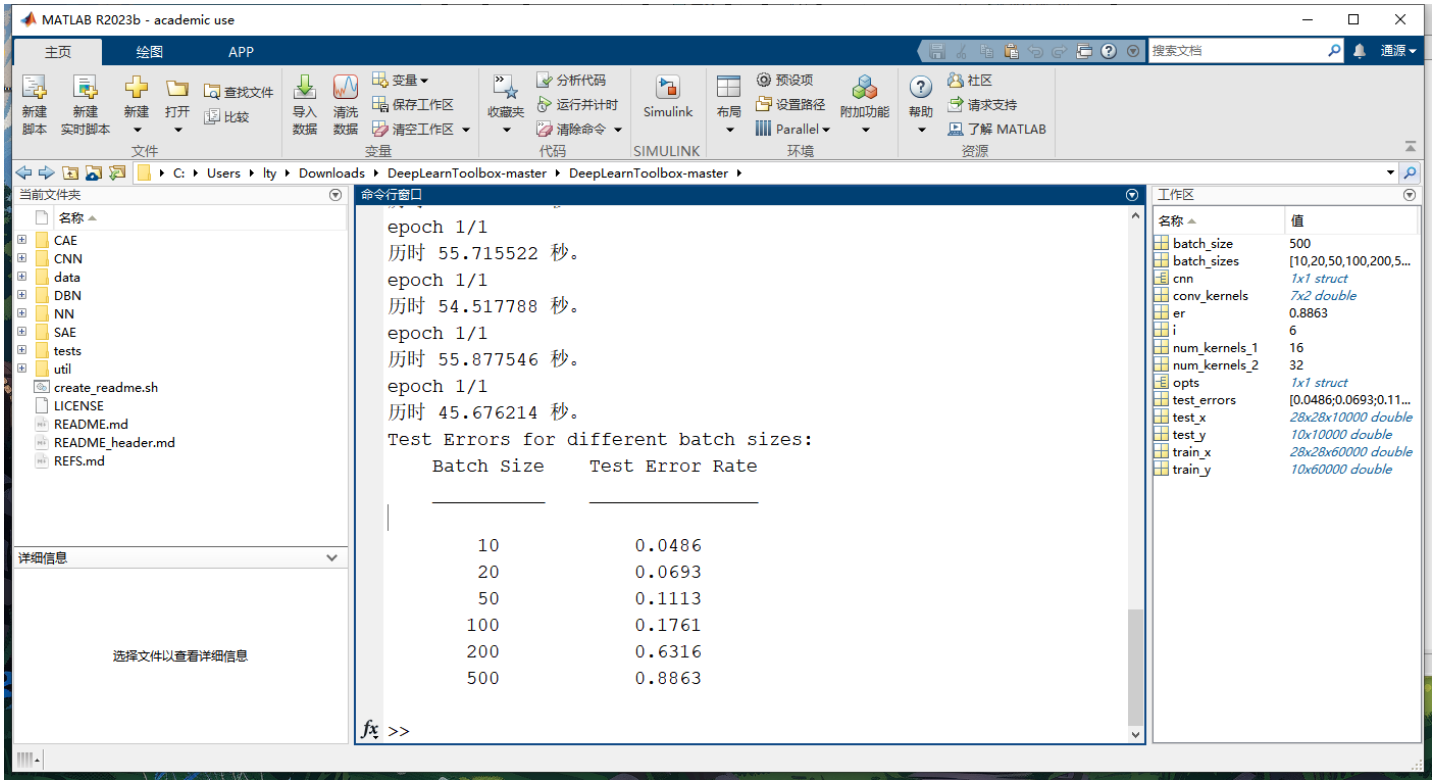
2

代码：

```
load mnist_uint8;

train_x = double(reshape(train_x', 28, 28, 60000)) / 255;
test_x = double(reshape(test_x', 28, 28, 10000)) / 255;
train_y = double(train_y');
test_y = double(test_y');

opts.alpha = 1;
opts.numepochs = 1;

% 定义不同的批量大小
batch_sizes = [10, 20, 50, 100, 200, 500];
test_errors = zeros(length(batch_sizes), 1); % 存储每种配置下的测试错误率

for i = 1:length(batch_sizes)
    batch_size = batch_sizes(i);
    opts.batchsize = batch_size;

```

```
    rand('state', 0)
    cnn.layers = {
        struct('type', 'i') % 输入层
        struct('type', 'c', 'outputmaps', 6, 'kernelsize', 5) % 第一个卷积层
        struct('type', 's', 'scale', 2) % 第一个子采样层
        struct('type', 'c', 'outputmaps', 12, 'kernelsize', 5) % 第二个卷积层
        struct('type', 's', 'scale', 2) % 第二个子采采层
    };
    cnn = cnnsetup(cnn, train_x, train_y);

    cnn = cnntrain(cnn, train_x, train_y, opts);

    [er, ~] = cnntest(cnn, test_x, test_y);
    test_errors(i) = er;
end

% 绘制批量大小与测试错误率的关系图
figure;
plot(batch_sizes, test_errors, '-o');
xlabel('Batch Size');
ylabel('Test Error Rate');
title('Test Error Rate vs Batch Size');
grid on;

disp('Test Errors for different batch sizes:');
disp(table(batch_sizes', test_errors, 'VariableNames', {'Batch Size', 'Test Error Rate'}));
```

结果：

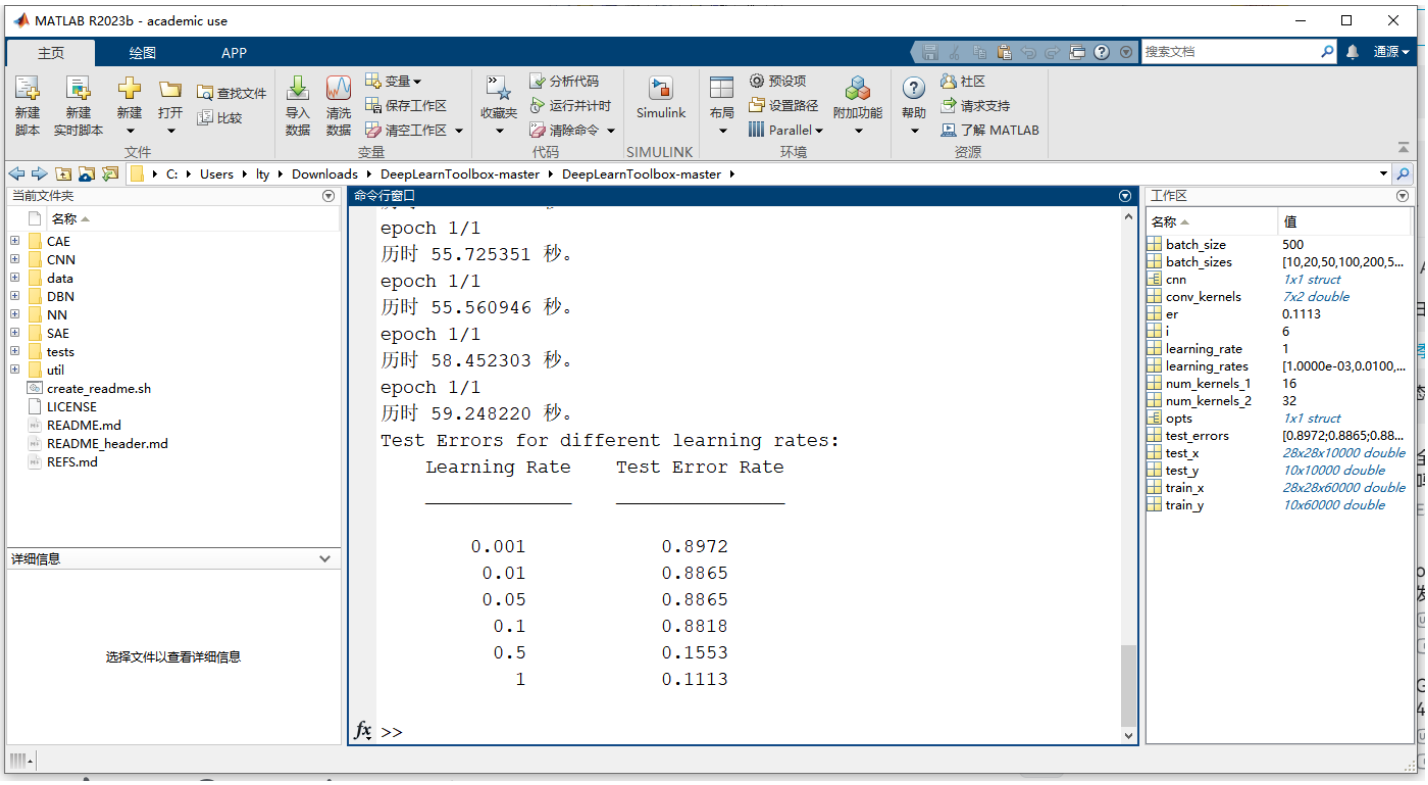


3

代码：

```
load mnist_uint8;

train_x = double(reshape(train_x', 28, 28, 60000)) / 255;
test_x = double(reshape(test_x', 28, 28, 10000)) / 255;
train_y = double(train_y');
test_y = double(test_y');

opts.batchsize = 50;
opts.numepochs = 1;

% 定义不同的学习率
learning_rates = [0.001, 0.01, 0.05, 0.1, 0.5, 1];
test_errors = zeros(length(learning_rates), 1); % 存储每种配置下的测试错误率

for i = 1:length(learning_rates)
    learning_rate = learning_rates(i);
    opts.alpha = learning_rate;

    rand('state', 0)
    cnn.layers = {
        struct('type', 'i') % 输入层
        struct('type', 'c', 'outputmaps', 6, 'kernelsize', 5) % 第一个卷积层
        struct('type', 's', 'scale', 2) % 第一个子采样层
        struct('type', 'c', 'outputmaps', 12, 'kernelsize', 5) % 第二个卷积层
        struct('type', 's', 'scale', 2) % 第二个子采样层
    };
    cnn = cnnsetup(cnn, train_x, train_y);

    cnn = cnntrain(cnn, train_x, train_y, opts);

    [er, ~] = cnntest(cnn, test_x, test_y);
    test_errors(i) = er;
end

% 绘制学习率与测试错误率的关系图
figure;
plot(learning_rates, test_errors, '-o');
xlabel('Learning Rate');
ylabel('Test Error Rate');
title('Test Error Rate vs Learning Rate');
grid on;

disp('Test Errors for different learning rates:');
disp(table(learning_rates', test_errors, 'VariableNames', {'Learning Rate',
'Test Error Rate'}));

```

结果：



4

代码：

```
load mnist_uint8;

train_x = double(reshape(train_x', 28, 28, 60000)) / 255;
test_x = double(reshape(test_x', 28, 28, 10000)) / 255;
train_y = double(train_y');
test_y = double(test_y');

opts.alpha = 1;
opts.batchsize = 50;

% 定义不同的训练轮数
num_epochs = [1, 3, 5, 7];
test_errors = zeros(length(num_epochs), 1); % 存储每种配置下的测试错误率

for i = 1:length(num_epochs)
    opts.numepochs = num_epochs(i);

    rand('state', 0)
    cnn.layers = {
        struct('type', 'i') % 输入层
        struct('type', 'c', 'outputmaps', 6, 'kernelsize', 5) % 第一个卷积层
        struct('type', 's', 'scale', 2) % 第一个子采样层
        struct('type', 'c', 'outputmaps', 12, 'kernelsize', 5) % 第二个卷积层
```

```
        struct('type', 's', 'scale', 2) % 第二个子采样层
    };
    cnn = cnnsetup(cnn, train_x, train_y);

    cnn = cnntrain(cnn, train_x, train_y, opts);

    [er, ~] = cnntest(cnn, test_x, test_y);
    test_errors(i) = er;
end

% 绘制训练轮数与测试错误率的关系图
figure;
plot(num_epochs, test_errors, '-o');
xlabel('Number of Epochs');
ylabel('Test Error Rate');
title('Test Error Rate vs Number of Epochs');
grid on;

disp('Test Errors for different number of epochs:');
disp(table(num_epochs', test_errors, 'VariableNames', {'Number of Epochs',
'Test Error Rate'}));

```

结果：