Neural-Network 3

Created time : 2024/5/24 09:49

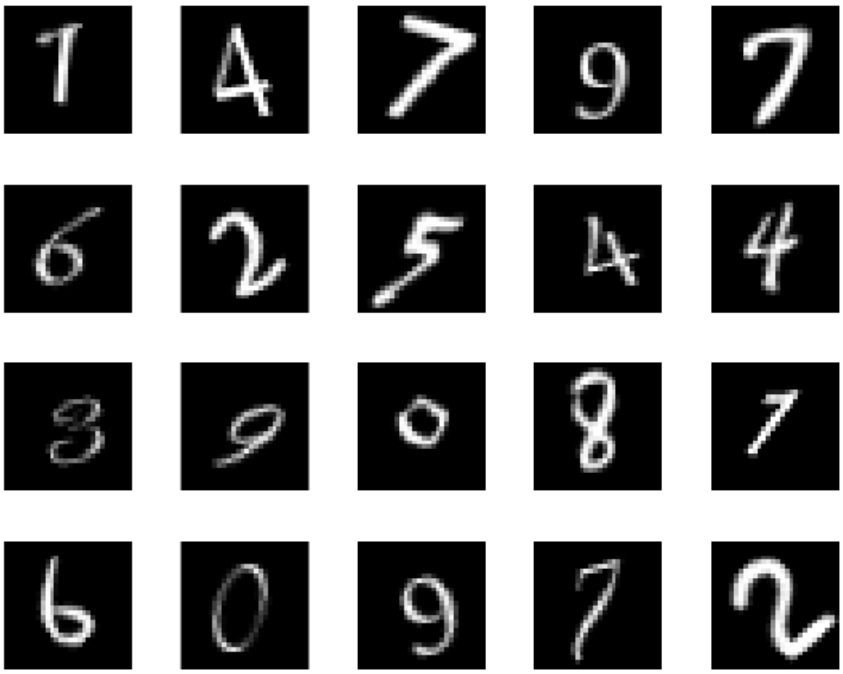
學號 : 109321019　姓名 : 涂价弘

```
clear
clc
```

CNN

步骤 1：加载和显示图像数据

```
[XTrain,~,YTrain] = digitTrain4DArrayData;                    %加载训练图像样本
[XValidation,~,YValidation] = digitTest4DArrayData;           %加载验证图像样本

% 随机显示 20 幅训练图像
numTrainImages = numel(YTrain);                               %统计用于训练样本的
数量
figure
idx = randperm(numTrainImages,20);
for i = 1:numel(idx)
    subplot(4,5,i)
    imshow(XTrain(:,:,:,idx(i)))
    drawnow
end
```



步骤 2:构建卷积神经网络

```
layers = [
    imageInputLayer([28 28 1])                                  % 输入层，1 个通道，像
素为 28×28

    convolution2dLayer(3,8,'Padding','same')                    % 卷积层 1：卷积核大
小为 3×3，卷积核的个数为 8（每个卷积核的通道数与输入图像的通道数相等，本层中每个卷积核 1 个通
道）卷积的方式采用零填充方式（即设定为 same 方式）
    batchNormalizationLayer                                     % 批量归一化层 1
    reluLayer                                                   % ReLU 非线性激活函数 1
    averagePooling2dLayer(2,'Stride',2)                         % 池化层 1：池化方式：
平均池化；池化区域为 2×2，步长为 2

    convolution2dLayer(3,16,'Padding','same')                   % 卷积层 2：卷积核大小
为 3×3，卷积核的个数为 16（每个卷积核的通道数与输入特征图的通道数相等，本层中每个卷积核 8 个
通道）卷积的方式采用零填充方式（即设定为 same 方式）
    batchNormalizationLayer                                     % 批量归一化层 2
    reluLayer                                                   % ReLU 非线性激活函数 2
    averagePooling2dLayer(2,'Stride',2)                         % 池化层 2：池化方式：
平均池化；池化区域为 2×2，步长为 2

    convolution2dLayer(3,32,'Padding','same')                   % 卷积层 3：卷积核大小
为 3×3，卷积核的个数为 32（每个卷积核的通道数与输入特征图的通道数相等，本层中每个卷积核 16 个
通道）卷积的方式采用零填充方式（即设定为 same 方式）
    batchNormalizationLayer                                     % 批量归一化层 3
```

```
    reluLayer                                                   % ReLU 非线性激活函数 3

    % convolution2dLayer(3,64,'Padding','same')
    % batchNormalizationLayer
    % reluLayer

    dropoutLayer(0.2)                                           % dropout 层，随机将 20%
的输入置零
    fullyConnectedLayer(1)                                      % 全连接层,全连接层的输
出为 1
    regressionLayer ];                                          % 回归层，用于预测结果
```

步骤 3：配置训练选项

```
miniBatchSize  = 128;                                           % 训练一次最小的样本量为
128
validationFrequency = floor(numel(YTrain)/miniBatchSize);       % 验证频率
options = trainingOptions('sgdm', ...                           % 设置训练方法，本例中将
其设置为 SGDM 法
    'MiniBatchSize',miniBatchSize, ...                          % 设置最小样本训练数量，
本例中将其设置为 128
    'MaxEpochs',30, ...                                         % 设置最大训练轮数，在本
例当中，最大训练轮数为 30
    'InitialLearnRate',0.001, ...                               % 设置初始学习率为 0.001
    'LearnRateSchedule','piecewise', ...                        % 设置初始的学习率是变化
的
    'LearnRateDropFactor',0.1, ...                              % 设置学习率衰减因子为
0.1
    'LearnRateDropPeriod',20, ...                               % 设置学习率衰减周期为
20 轮，即：每 20 轮，在之前的学习率基础上，乘以学习率的衰减因子 0.1
    'Shuffle','every-epoch', ...                                % 设置每一轮都打乱数据
    'ValidationData',{XValidation,YValidation}, ...             % 设置验证用得数据
    'ValidationFrequency',validationFrequency, ...              % 设置验证频率
    'Plots','training-progress', ...                            % 设置打开训练进度图
    'Verbose',true);                                           % 设置关闭命令窗口的输出
```

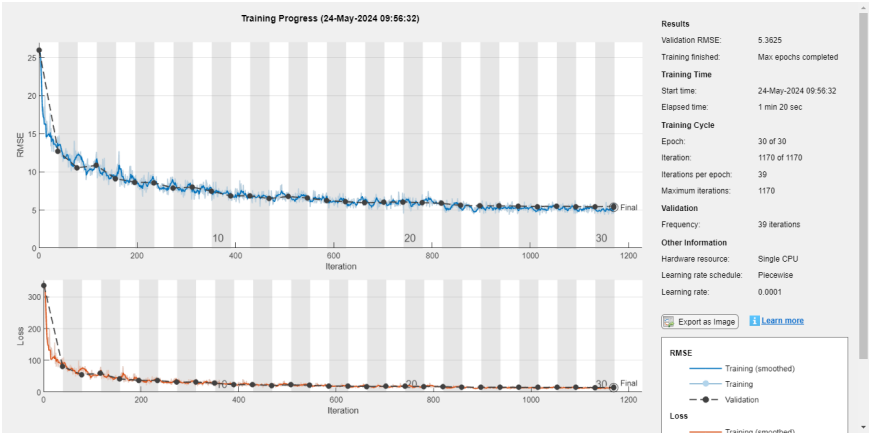
步骤 4：训练网络

```
net = trainNetwork(XTrain,YTrain,layers,options);
```

```
Training on single CPU.
Initializing input data normalization.
|=========================================================================================================================
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Validation  |  Mini-batch  |  Validation  |  Base Learnin
|         |             |   (hh:mm:ss)   |     RMSE     |     RMSE     |     Loss     |     Loss     |      Rate
|=========================================================================================================================
|       1 |           1 |       00:00:04 |        25.71 |        25.94 |     330.5157 |     336.3932 |          0.00
|       1 |          39 |       00:00:07 |        12.69 |        12.73 |      80.4754 |      80.9785 |          0.00
|       2 |          50 |       00:00:08 |        11.89 |              |      70.6410 |              |          0.00
|       2 |          78 |       00:00:10 |        11.61 |        10.53 |      67.3771 |      55.4436 |          0.00
|       3 |         100 |       00:00:11 |         9.88 |              |      48.8142 |              |          0.00
|       3 |         117 |       00:00:12 |        12.56 |        10.84 |      78.8615 |      58.7907 |          0.00
```

4	150	00:00:14	9.13		41.6857		0.00
4	156	00:00:15	9.38	9.10	44.0328	41.4459	0.00
5	195	00:00:17	8.90	8.64	39.6065	37.3258	0.00
6	200	00:00:18	8.89		39.4718		0.00
6	234	00:00:20	8.46	8.57	35.8233	36.7390	0.00
7	250	00:00:21	8.42		35.4250		0.00
7	273	00:00:22	7.35	7.89	27.0476	31.0910	0.00
8	300	00:00:24	7.81		30.4695		0.00
8	312	00:00:25	7.13	8.00	25.4220	31.9616	0.00
9	350	00:00:27	7.26		26.3340		0.00
9	351	00:00:27	7.55	7.46	28.5124	27.8383	0.00
10	390	00:00:30	7.61	6.84	28.9519	23.4254	0.00
11	400	00:00:30	7.79		30.3657		0.00
11	429	00:00:32	7.17	6.85	25.6873	23.4294	0.00
12	450	00:00:33	5.48		14.9889		0.00
12	468	00:00:35	7.57	6.49	28.6854	21.0423	0.00
13	500	00:00:37	6.36		20.2319		0.00
13	507	00:00:38	7.31	6.77	26.7461	22.8970	0.00
14	546	00:00:40	6.63	6.56	21.9793	21.4978	0.00
15	550	00:00:40	6.74		22.7126		0.00
15	585	00:00:42	7.22	6.24	26.0517	19.4712	0.00
16	600	00:00:43	5.75		16.5569		0.00
16	624	00:00:45	6.75	6.07	22.7604	18.4367	0.00
17	650	00:00:46	6.53		21.3216		0.00
17	663	00:00:47	6.47	5.95	20.9372	17.7266	0.00
18	700	00:00:49	6.07		18.3959		0.00
18	702	00:00:50	6.57	6.01	21.5805	18.0665	0.00
19	741	00:00:52	6.13	6.05	18.7824	18.2884	0.00
20	750	00:00:52	5.97		17.8419		0.00
20	780	00:00:54	6.98	5.95	24.3458	17.6987	0.00
21	800	00:00:55	5.57		15.5143		0.00
21	819	00:00:57	6.21	5.87	19.3032	17.2440	0.00
22	850	00:00:58	5.35		14.3181		0.00
22	858	00:00:59	5.30	5.59	14.0610	15.5986	0.00
23	897	00:01:02	4.67	5.54	10.9269	15.3229	0.00
24	900	00:01:02	5.00		12.5112		0.00
24	936	00:01:04	4.67	5.53	10.8831	15.2916	0.00
25	950	00:01:05	5.09		12.9509		0.00
25	975	00:01:07	4.83	5.50	11.6763	15.1468	0.00
26	1000	00:01:08	5.53		15.3140		0.00
26	1014	00:01:09	5.13	5.43	13.1372	14.7688	0.00
27	1050	00:01:12	5.77		16.6697		0.00
27	1053	00:01:12	5.02	5.51	12.6071	15.1588	0.00
28	1092	00:01:15	4.55	5.44	10.3384	14.7708	0.00
29	1100	00:01:15	5.41		14.6406		0.00
29	1131	00:01:17	4.58	5.41	10.4959	14.6412	0.00
30	1150	00:01:18	4.61		10.6429		0.00
30	1170	00:01:20	5.53	5.46	15.2819	14.9105	0.00

Training finished: Max epochs completed.



步骤 5：测试与评估

```
YPredicted = predict(net,XValidation);                    % 用训练好的网络预测验证图像中数字倾斜的角度
predictionError = YValidation - YPredicted;               % 计算预测倾斜角度和实际倾斜角度之间的预测误差
% 计算准确率
thr = 10;                                                 % 设定阈值，在本例中，阈值设定为 10 度
numCorrect = sum(abs(predictionError) < thr);             % 当预测值与实际值得误差小于 10 度时，则认为预测正确
numValidationImages = numel(YValidation);                 % 用于验证图像的数量
Accuracy = numCorrect/numValidationImages                 % 计算准确率
```

```
Accuracy = 0.9412
```

```
% 计算 RMSE 的值
squares = predictionError.^2;
RMSE = sqrt(mean(squares))
```

```
RMSE = single

    5.3625
```

```
% original layers

layers = [
    imageInputLayer([28 28 1])

    convolution2dLayer(3,8,'Padding','same')
    batchNormalizationLayer
    reluLayer
    averagePooling2dLayer(2,'Stride',2)

    convolution2dLayer(3,16,'Padding','same')
    batchNormalizationLayer
    reluLayer
```

```
    averagePooling2dLayer(2,'Stride',2)

    convolution2dLayer(3,32,'Padding','same')
    batchNormalizationLayer
    reluLayer

    % convolution2dLayer(3,64,'Padding','same')
    % batchNormalizationLayer
    % reluLayer

    dropoutLayer(0.2)
    fullyConnectedLayer(1)
    regressionLayer ];
```

Exercise-1

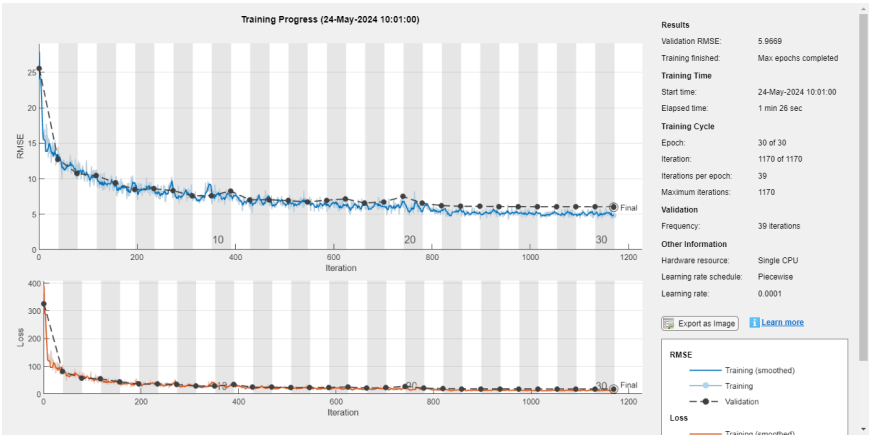
```
layers = [
    imageInputLayer([28 28 1])

    convolution2dLayer(3,8,'Padding','same')
    batchNormalizationLayer
    reluLayer
    maxPooling2dLayer(2,'Stride',2)

    convolution2dLayer(3,16,'Padding','same')
    batchNormalizationLayer
    reluLayer
    maxPooling2dLayer(2,'Stride',2)

    convolution2dLayer(3,32,'Padding','same')
    batchNormalizationLayer
    reluLayer

    dropoutLayer(0.2)
    fullyConnectedLayer(1)
    regressionLayer ];

net = trainNetwork(XTrain,YTrain,layers,options);
```

```
Training on single CPU.
Initializing input data normalization.
|=========================================================================================================================
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Validation  |  Mini-batch  |  Validation  |  Base Learnin
|         |             |   (hh:mm:ss)   |     RMSE     |     RMSE     |     Loss     |     Loss     |      Rate
|=========================================================================================================================
|       1 |           1 |       00:00:01 |        27.91 |        25.52 |     389.5237 |     325.5949 |          0.00
|       1 |          39 |       00:00:04 |        14.46 |        12.75 |     104.5097 |      81.2544 |          0.00
|       2 |          50 |       00:00:05 |        11.12 |              |      61.8182 |              |          0.00
|       2 |          78 |       00:00:07 |        11.53 |        10.71 |      66.4905 |      57.3150 |          0.00
|       3 |         100 |       00:00:08 |        10.46 |              |      54.7497 |              |          0.00
|       3 |         117 |       00:00:09 |        11.21 |        10.44 |      62.8011 |      54.4492 |          0.00
|       4 |         150 |       00:00:11 |         8.69 |              |      37.7529 |              |          0.00
|       4 |         156 |       00:00:12 |         8.93 |         9.40 |      39.8937 |      44.1950 |          0.00
```

5	195	00:00:14	9.09	8.44	41.3444	35.6517	0.00
6	200	00:00:14	7.67		29.3953		0.00
6	234	00:00:17	7.88	8.58	31.0580	36.7924	0.00
7	250	00:00:17	7.82		30.5874		0.00
7	273	00:00:19	7.31	8.32	26.7448	34.6068	0.00
8	300	00:00:20	7.80		30.4361		0.00
8	312	00:00:22	7.21	7.58	26.0247	28.7218	0.00
9	350	00:00:23	8.70		37.8849		0.00
9	351	00:00:24	7.14	7.61	25.4649	28.9807	0.00
10	390	00:00:27	7.69	8.25	29.6010	34.0483	0.00
11	400	00:00:27	7.67		29.4209		0.00
11	429	00:00:29	6.93	7.02	23.9879	24.6090	0.00
12	450	00:00:30	6.43		20.6538		0.00
12	468	00:00:31	7.55	7.03	28.4640	24.7124	0.00
13	500	00:00:33	6.66		22.1825		0.00
13	507	00:00:34	6.58	6.90	21.6221	23.8010	0.00
14	546	00:00:36	7.01	6.73	24.5487	22.6336	0.00
15	550	00:00:37	6.38		20.3466		0.00
15	585	00:00:39	6.89	6.96	23.7222	24.2106	0.00
16	600	00:00:40	5.98		17.9032		0.00
16	624	00:00:42	6.27	7.14	19.6691	25.5227	0.00
17	650	00:00:44	7.05		24.8222		0.00
17	663	00:00:45	6.59	6.56	21.7399	21.5457	0.00
18	700	00:00:48	5.61		15.7296		0.00
18	702	00:00:49	6.16	6.73	18.9559	22.6574	0.00
19	741	00:00:52	5.96	7.48	17.7527	27.9607	0.00
20	750	00:00:53	6.01		18.0340		0.00
20	780	00:00:56	4.93	6.54	12.1673	21.3640	0.00
21	800	00:00:57	5.89		17.3486		0.00
21	819	00:01:00	5.49	6.20	15.0578	19.2367	0.00
22	850	00:01:01	4.96		12.2862		0.00
22	858	00:01:02	5.53	6.11	15.2984	18.6492	0.00
23	897	00:01:07	5.34	6.11	14.2749	18.6776	0.00
24	900	00:01:07	4.92		12.0949		0.00
24	936	00:01:11	5.31	6.08	14.0858	18.4905	0.00
25	950	00:01:11	5.25		13.7737		0.00
25	975	00:01:13	4.91	6.07	12.0502	18.4513	0.00
26	1000	00:01:14	4.94		12.2050		0.00
26	1014	00:01:16	4.83	6.05	11.6499	18.3302	0.00
27	1050	00:01:18	4.90		12.0241		0.00
27	1053	00:01:18	5.02	6.05	12.6095	18.3141	0.00
28	1092	00:01:21	4.95	6.06	12.2440	18.3425	0.00
29	1100	00:01:21	5.13		13.1725		0.00
29	1131	00:01:23	5.35	6.05	14.2877	18.2724	0.00
30	1150	00:01:24	5.18		13.4208		0.00
30	1170	00:01:25	4.79	6.04	11.4873	18.2620	0.00

|==

Training finished: Max epochs completed.



```
YPredicted = predict(net,XValidation);
predictionError = YValidation - YPredicted;

% Accuracy
thr = 10;
numCorrect = sum(abs(predictionError) < thr);
numValidationImages = numel(YValidation);
Accuracy = numCorrect/numValidationImages
```

```
Accuracy = 0.9134
```

```
% RMSE
squares = predictionError.^2;
RMSE = sqrt(mean(squares))
```

```
RMSE = single

    5.9669
```

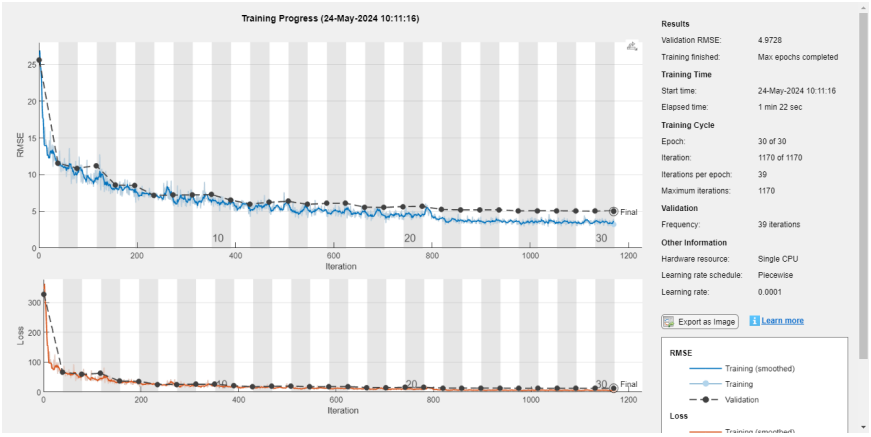
Exercise-2-1

```
% original layers

layers = [
    imageInputLayer([28 28 1])

    convolution2dLayer(3,8,'Padding','same')
    batchNormalizationLayer
    reluLayer
    averagePooling2dLayer(2,'Stride',2)

    convolution2dLayer(3,16,'Padding','same')
    batchNormalizationLayer
    reluLayer
    averagePooling2dLayer(2,'Stride',2)

    convolution2dLayer(3,32,'Padding','same')
    batchNormalizationLayer
    reluLayer

    fullyConnectedLayer(1)
    regressionLayer ];

net = trainNetwork(XTrain,YTrain,layers,options);
```

```
Training on single CPU.
Initializing input data normalization.
|=========================================================================================================================
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Validation  |  Mini-batch  |  Validation  |  Base Learnin
|         |             |   (hh:mm:ss)   |     RMSE     |     RMSE     |     Loss     |     Loss     |      Rate
```

1	1	00:00:02	26.26	25.60	344.7772	327.5879	0.00
1	39	00:00:04	11.61	11.56	67.4399	66.8226	0.00
2	50	00:00:05	11.15		62.2051		0.00
2	78	00:00:07	10.66	10.87	56.8090	59.0897	0.00
3	100	00:00:08	9.96		49.6225		0.00
3	117	00:00:09	8.88	11.22	39.4285	62.9812	0.00
4	150	00:00:11	7.55		28.5314		0.00
4	156	00:00:12	8.19	8.57	33.5246	36.7574	0.00
5	195	00:00:14	8.88	8.52	39.4113	36.3073	0.00
6	200	00:00:14	6.94		24.1097		0.00
6	234	00:00:17	7.30	7.17	26.6656	25.7158	0.00
7	250	00:00:17	6.79		23.0240		0.00
7	273	00:00:19	6.25	7.21	19.5510	26.0150	0.00
8	300	00:00:20	5.68		16.1111		0.00
8	312	00:00:22	6.13	7.24	18.7926	26.2228	0.00
9	350	00:00:23	7.06		24.9177		0.00
9	351	00:00:24	5.85	7.28	17.0999	26.4969	0.00
10	390	00:00:27	5.98	6.54	17.8768	21.3955	0.00
11	400	00:00:28	5.76		16.5753		0.00
11	429	00:00:30	5.04	6.00	12.6831	17.9974	0.00
12	450	00:00:31	5.83		16.9859		0.00
12	468	00:00:33	6.72	6.22	22.5877	19.3341	0.00
13	500	00:00:34	5.58		15.5650		0.00
13	507	00:00:35	5.07	6.37	12.8554	20.2578	0.00
14	546	00:00:38	5.99	5.96	17.9679	17.7471	0.00
15	550	00:00:39	5.06		12.8144		0.00
15	585	00:00:41	4.82	6.08	11.6303	18.4599	0.00
16	600	00:00:42	5.44		14.7753		0.00
16	624	00:00:44	4.98	6.10	12.4097	18.5864	0.00
17	650	00:00:45	4.58		10.4832		0.00
17	663	00:00:46	5.61	5.57	15.7428	15.5022	0.00
18	700	00:00:48	4.68		10.9499		0.00
18	702	00:00:49	4.75	5.53	11.2760	15.2649	0.00
19	741	00:00:52	4.23	5.65	8.9347	15.9555	0.00
20	750	00:00:52	4.34		9.4267		0.00
20	780	00:00:55	5.08	5.66	12.8957	16.0294	0.00
21	800	00:00:56	4.10		8.4126		0.00
21	819	00:00:57	4.13	5.26	8.5342	13.8111	0.00
22	850	00:00:59	3.45		5.9611		0.00
22	858	00:01:00	3.89	5.20	7.5517	13.5407	0.00
23	897	00:01:02	3.61	5.20	6.5054	13.5114	0.00
24	900	00:01:03	3.25		5.2899		0.00
24	936	00:01:05	3.84	5.21	7.3568	13.5709	0.00
25	950	00:01:06	3.17		5.0254		0.00
25	975	00:01:08	3.22	5.06	5.1727	12.7933	0.00
26	1000	00:01:09	4.01		8.0203		0.00
26	1014	00:01:10	3.81	5.08	7.2533	12.9071	0.00
27	1050	00:01:12	3.31		5.4799		0.00
27	1053	00:01:13	3.77	5.08	7.1042	12.8976	0.00
28	1092	00:01:16	3.67	5.04	6.7505	12.6832	0.00
29	1100	00:01:17	3.56		6.3458		0.00
29	1131	00:01:19	3.19	5.03	5.0810	12.6340	0.00
30	1150	00:01:20	3.35		5.6184		0.00
30	1170	00:01:22	3.24	5.05	5.2426	12.7462	0.00

Training finished: Max epochs completed.




```
YPredicted = predict(net,XValidation);
predictionError = YValidation - YPredicted;

% Accuracy
thr = 10;
numCorrect = sum(abs(predictionError) < thr);
numValidationImages = numel(YValidation);
Accuracy = numCorrect/numValidationImages
```

```
Accuracy = 0.9554
```

```
% RMSE
squares = predictionError.^2;
RMSE = sqrt(mean(squares))
```

```
RMSE = single

    4.9728
```

Exercise-2-2

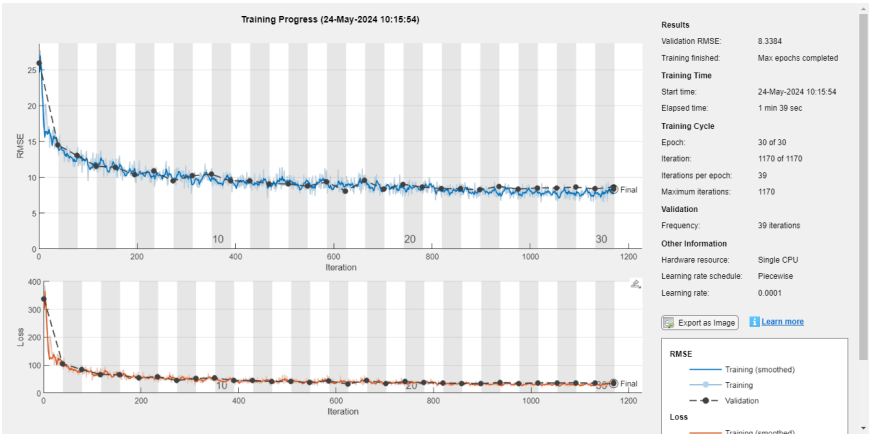
```
layers = [
    imageInputLayer([28 28 1])

    convolution2dLayer(3,8,'Padding','same')
    dropoutLayer(0.2)
    batchNormalizationLayer
    reluLayer
    averagePooling2dLayer(2,'Stride',2)

    convolution2dLayer(3,16,'Padding','same')
    dropoutLayer(0.2)
    batchNormalizationLayer
    reluLayer
    averagePooling2dLayer(2,'Stride',2)
```

```
    convolution2dLayer(3,32,'Padding','same')
    dropoutLayer(0.2)
    batchNormalizationLayer
    reluLayer

    dropoutLayer(0.2)
    fullyConnectedLayer(1)
    regressionLayer ];

net = trainNetwork(XTrain,YTrain,layers,options);
```

```
Training on single CPU.
Initializing input data normalization.
```

Epoch	Iteration	Time Elapsed (hh:mm:ss)	Mini-batch RMSE	Validation RMSE	Mini-batch Loss	Validation Loss	Base Learnin Rate
1	1	00:00:02	26.03	25.99	338.8383	337.8305	0.00
1	39	00:00:05	15.34	14.52	117.6599	105.4386	0.00
2	50	00:00:06	13.82		95.5306		0.00
2	78	00:00:08	13.33	13.04	88.7901	85.0469	0.00
3	100	00:00:10	9.57		45.8172		0.00
3	117	00:00:12	13.26	11.57	87.8583	66.8865	0.00
4	150	00:00:14	12.57		78.9978		0.00
4	156	00:00:15	11.59	11.42	67.1797	65.2398	0.00
5	195	00:00:18	11.34	10.41	64.2462	54.1841	0.00
6	200	00:00:18	10.31		53.1459		0.00
6	234	00:00:21	11.26	10.88	63.3883	59.2030	0.00
7	250	00:00:22	10.22		52.2519		0.00
7	273	00:00:25	9.50	9.52	45.1600	45.3050	0.00
8	300	00:00:27	9.28		43.0326		0.00
8	312	00:00:28	9.53	10.25	45.3709	52.4819	0.00
9	350	00:00:31	9.83		48.3170		0.00
9	351	00:00:31	11.03	10.45	60.8482	54.6026	0.00
10	390	00:00:34	10.53	9.52	55.4845	45.2752	0.00
11	400	00:00:35	9.64		46.4252		0.00
11	429	00:00:38	10.05	9.55	50.5329	45.6439	0.00
12	450	00:00:39	9.77		47.7457		0.00
12	468	00:00:41	10.19	9.11	51.8825	41.5126	0.00
13	500	00:00:43	8.93		39.8418		0.00
13	507	00:00:44	9.64	9.10	46.4686	41.4105	0.00
14	546	00:00:47	9.52	8.77	45.2689	38.4273	0.00
15	550	00:00:47	9.91		49.1074		0.00
15	585	00:00:50	10.93	9.35	59.7264	43.7522	0.00
16	600	00:00:51	8.64		37.3574		0.00
16	624	00:00:53	9.19	8.11	42.1839	32.9114	0.00
17	650	00:00:55	9.15		41.8294		0.00
17	663	00:00:57	8.84	9.57	39.0641	45.8177	0.00
18	700	00:00:59	8.49		36.0080		0.00
18	702	00:01:00	8.20	8.37	33.5860	35.0376	0.00
19	741	00:01:03	8.89	9.05	39.5468	40.9450	0.00
20	750	00:01:04	8.15		33.1710		0.00
20	780	00:01:06	8.22	8.66	33.8028	37.4560	0.00
21	800	00:01:08	8.05		32.3938		0.00
21	819	00:01:09	7.24	8.45	26.1915	35.6955	0.00
22	850	00:01:11	8.99		40.3654		0.00
22	858	00:01:13	7.61	8.40	28.9659	35.3049	0.00
23	897	00:01:16	7.40	8.28	27.3941	34.2874	0.00
24	900	00:01:16	7.98		31.8199		0.00
24	936	00:01:19	7.56	8.75	28.5987	38.3049	0.00
25	950	00:01:20	7.67		29.4343		0.00
25	975	00:01:22	6.96	8.38	24.2266	35.1058	0.00

```
|      26 |        1000 |       00:01:24 |         7.40 |              |      27.3831 |              |          0.00
|      26 |        1014 |       00:01:25 |         7.41 |         8.53 |      27.4897 |      36.4050 |          0.00
|      27 |        1050 |       00:01:28 |         8.28 |              |      34.2524 |              |          0.00
|      27 |        1053 |       00:01:29 |         7.61 |         8.54 |      28.9501 |      36.4979 |          0.00
|      28 |        1092 |       00:01:32 |         7.50 |         8.63 |      28.1438 |      37.1994 |          0.00
|      29 |        1100 |       00:01:32 |         6.75 |              |      22.7491 |              |          0.00
|      29 |        1131 |       00:01:35 |         8.80 |         8.42 |      38.7179 |      35.4350 |          0.00
|      30 |        1150 |       00:01:37 |         8.03 |              |      32.2269 |              |          0.00
|      30 |        1170 |       00:01:38 |         8.42 |         8.63 |      35.4597 |      37.2592 |          0.00
|=========================================================================================================================================
Training finished: Max epochs completed.
```




```
YPredicted = predict(net,XValidation);
predictionError = YValidation - YPredicted;

% Accuracy
thr = 10;
numCorrect = sum(abs(predictionError) < thr);
numValidationImages = numel(YValidation);
Accuracy = numCorrect/numValidationImages
```

```
Accuracy = 0.7804
```

```
% RMSE
squares = predictionError.^2;
RMSE = sqrt(mean(squares))
```

```
RMSE = single

    8.3384
```

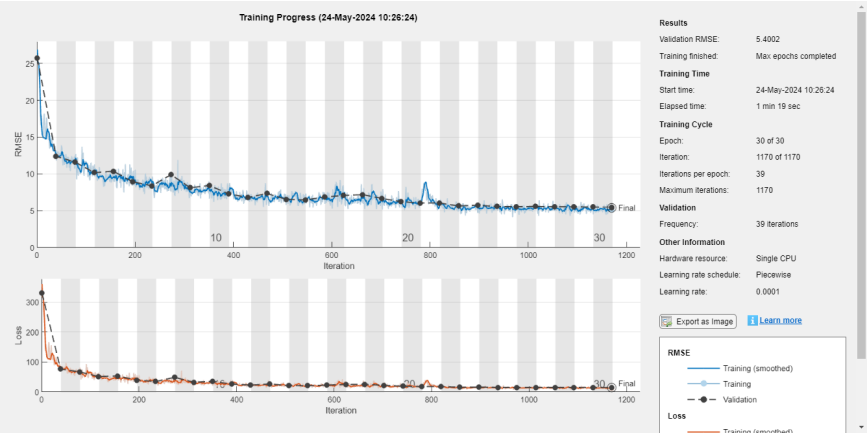
Exercise-3

```
layers = [
    imageInputLayer([28 28 1])

    convolution2dLayer(3,8,'Padding','same')
    batchNormalizationLayer
    reluLayer
    averagePooling2dLayer(2,'Stride',2)
```

```
    convolution2dLayer(3,16,'Padding','same')
    batchNormalizationLayer
    reluLayer
    averagePooling2dLayer(2,'Stride',2)

    convolution2dLayer(3,32,'Padding','same')
    batchNormalizationLayer
    reluLayer

    dropoutLayer(0.2)
    fullyConnectedLayer(1)
    regressionLayer ];

net = trainNetwork(XTrain,YTrain,layers,options);
```

```
Training on single CPU.
Initializing input data normalization.
|=========================================================================================================================
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Validation  |  Mini-batch  |  Validation  |  Base Learnin
|         |             |   (hh:mm:ss)   |     RMSE     |     RMSE     |     Loss     |     Loss     |      Rate
|=========================================================================================================================
|       1 |           1 |       00:00:02 |        26.87 |        25.72 |     361.0811 |     330.6974 |          0.00
|       1 |          39 |       00:00:05 |        13.15 |        12.40 |      86.4637 |      76.8540 |          0.00
|       2 |          50 |       00:00:06 |        13.83 |              |      95.6960 |              |          0.00
|       2 |          78 |       00:00:08 |        12.37 |        11.59 |      76.4768 |      67.2011 |          0.00
|       3 |         100 |       00:00:09 |        11.68 |              |      68.2300 |              |          0.00
|       3 |         117 |       00:00:10 |        11.20 |        10.17 |      62.7307 |      51.7025 |          0.00
|       4 |         150 |       00:00:12 |         9.36 |              |      43.8416 |              |          0.00
|       4 |         156 |       00:00:13 |         9.04 |        10.36 |      40.8762 |      53.6665 |          0.00
|       5 |         195 |       00:00:15 |         8.84 |         8.93 |      39.0727 |      39.8328 |          0.00
|       6 |         200 |       00:00:16 |        10.49 |              |      54.9870 |              |          0.00
|       6 |         234 |       00:00:18 |         8.50 |         8.39 |      36.1345 |      35.1804 |          0.00
|       7 |         250 |       00:00:19 |         7.95 |              |      31.6121 |              |          0.00
|       7 |         273 |       00:00:20 |         7.66 |         9.90 |      29.3061 |      49.0069 |          0.00
|       8 |         300 |       00:00:22 |         7.82 |              |      30.5979 |              |          0.00
|       8 |         312 |       00:00:23 |         8.43 |         8.12 |      35.5713 |      33.0058 |          0.00
|       9 |         350 |       00:00:25 |         8.96 |              |      40.1335 |              |          0.00
|       9 |         351 |       00:00:25 |         6.68 |         8.43 |      22.2973 |      35.5135 |          0.00
|      10 |         390 |       00:00:28 |         7.95 |         7.30 |      31.5895 |      26.6581 |          0.00
|      11 |         400 |       00:00:28 |         6.87 |              |      23.6069 |              |          0.00
|      11 |         429 |       00:00:31 |         6.51 |         6.80 |      21.1873 |      23.1398 |          0.00
|      12 |         450 |       00:00:32 |         6.72 |              |      22.5613 |              |          0.00
|      12 |         468 |       00:00:33 |         6.75 |         7.34 |      22.7642 |      26.9263 |          0.00
|      13 |         500 |       00:00:35 |         6.62 |              |      21.9131 |              |          0.00
|      13 |         507 |       00:00:35 |         5.83 |         6.56 |      17.0203 |      21.5157 |          0.00
|      14 |         546 |       00:00:38 |         6.44 |         6.46 |      20.7363 |      20.8938 |          0.00
|      15 |         550 |       00:00:38 |         6.59 |              |      21.7458 |              |          0.00
|      15 |         585 |       00:00:40 |         7.41 |         6.85 |      27.4344 |      23.4858 |          0.00
|      16 |         600 |       00:00:41 |         6.35 |              |      20.1453 |              |          0.00
|      16 |         624 |       00:00:43 |         7.58 |         7.08 |      28.7430 |      25.0339 |          0.00
|      17 |         650 |       00:00:44 |         6.68 |              |      22.3117 |              |          0.00
|      17 |         663 |       00:00:45 |         7.23 |         7.18 |      26.1559 |      25.8111 |          0.00
|      18 |         700 |       00:00:47 |         7.56 |              |      28.5721 |              |          0.00
|      18 |         702 |       00:00:48 |         5.85 |         6.67 |      17.1074 |      22.2536 |          0.00
|      19 |         741 |       00:00:51 |         6.18 |         6.23 |      19.1108 |      19.4216 |          0.00
|      20 |         750 |       00:00:51 |         7.15 |              |      25.5635 |              |          0.00
|      20 |         780 |       00:00:53 |         6.50 |         6.05 |      21.1110 |      18.3045 |          0.00
|      21 |         800 |       00:00:54 |         7.28 |              |      26.4830 |              |          0.00
|      21 |         819 |       00:00:56 |         6.42 |         6.05 |      20.5774 |      18.3001 |          0.00
```

```
|      22 |         850 |       00:00:57 |         5.37 |              |      14.4162 |              |          0.00
|      22 |         858 |       00:00:58 |         5.00 |         5.65 |      12.5138 |      15.9663 |          0.00
|      23 |         897 |       00:01:01 |         4.74 |         5.75 |      11.2454 |      16.5182 |          0.00
|      24 |         900 |       00:01:01 |         4.68 |              |      10.9293 |              |          0.00
|      24 |         936 |       00:01:03 |         5.78 |         5.58 |      16.6981 |      15.5465 |          0.00
|      25 |         950 |       00:01:04 |         5.59 |              |      15.6256 |              |          0.00
|      25 |         975 |       00:01:06 |         5.61 |         5.55 |      15.7288 |      15.4235 |          0.00
|      26 |        1000 |       00:01:07 |         5.45 |              |      14.8361 |              |          0.00
|      26 |        1014 |       00:01:08 |         5.36 |         5.59 |      14.3592 |      15.6102 |          0.00
|      27 |        1050 |       00:01:10 |         6.31 |              |      19.9023 |              |          0.00
|      27 |        1053 |       00:01:11 |         5.03 |         5.55 |      12.6372 |      15.3736 |          0.00
|      28 |        1092 |       00:01:13 |         5.22 |         5.54 |      13.6115 |      15.3399 |          0.00
|      29 |        1100 |       00:01:14 |         4.83 |              |      11.6689 |              |          0.00
|      29 |        1131 |       00:01:16 |         5.54 |         5.50 |      15.3649 |      15.1521 |          0.00
|      30 |        1150 |       00:01:17 |         5.31 |              |      14.0737 |              |          0.00
|      30 |        1170 |       00:01:18 |         5.12 |         5.47 |      13.1260 |      14.9794 |          0.00
|=========================================================================================================================================
Training finished: Max epochs completed.
```




```
YPredicted = predict(net,XValidation);
predictionError = YValidation - YPredicted;

% Accuracy
thr = 10;
numCorrect = sum(abs(predictionError) < thr);
numValidationImages = numel(YValidation);
Accuracy = numCorrect/numValidationImages
```

```
Accuracy = 0.9408
```

```
% RMSE
squares = predictionError.^2;
RMSE = sqrt(mean(squares))
```

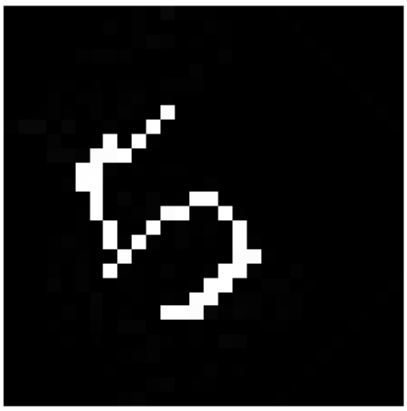
```
RMSE = single

    5.4002
```

```
clf('reset')

% test with real image
```

```
I = imread('./images/hand_write_5.png');
I45 = imrotate(I, 45, 'crop');
I45 = rgb2gray(I45)
```

```
I45 = 28×28 uint8 matrix
     0     0     0     0     0     0     0     0     0     2     0     7     1 ···
     0     0     0     0     0     0     0     3     0     2     0     0     0
     0     0     0     0     0     0     0     0     0     2     5     3     0
     0     0     0     0     0     2     3     3     2     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     1     1     0     1     0     1     1     0     2     2
     0     0     0     2     0     2     2     0     2     3     0     1     0
     0     0     1     2     1     0     3     0     3     1     0   255     0
     2     7     7     0     6     0     0     3     0     0   254     0     0
     0     0     0     4     0     0     1   254     0   255     1     0     0
     ⋮
```

```
imshow(I45);
truesize([200 200])
```



```
predict(net, double(I45)/255)
```

```
ans = single

    -19.2021
```

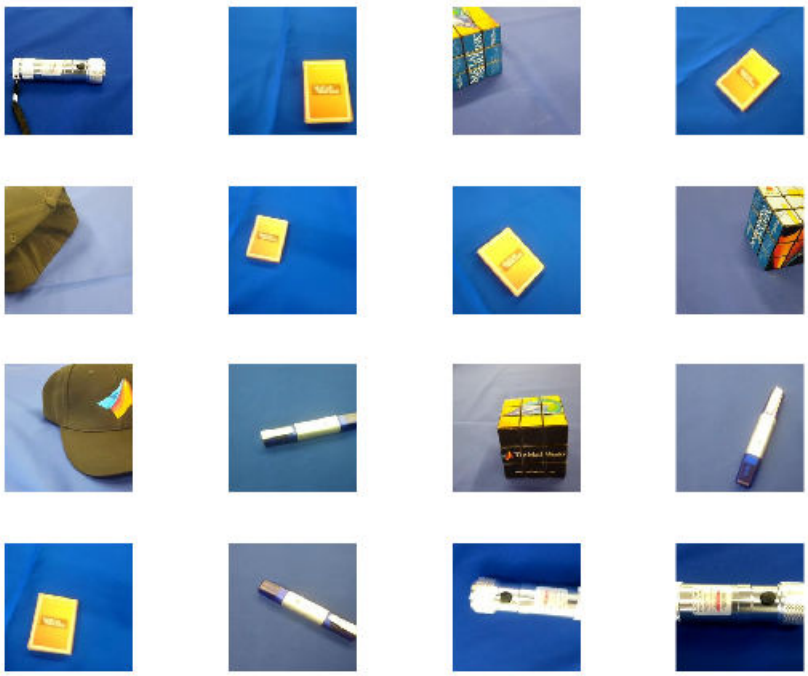
AlexNet with transfer learning

Exercise-4

步骤 1：加载图像数据，并将其划分为训练集和验证集

```
% 加载图像数据
unzip('./data/MerchData.zip');
imds = imageDatastore('MerchData', ...
    'IncludeSubfolders',true, ...
    'LabelSource','foldernames');
```

```
% 划分验证集和训练集
[imdsTrain,imdsValidation] = splitEachLabel(imds,0.7,'randomized');

% 随机显示训练集中的部分图像
numTrainImages = numel(imdsTrain.Labels);
idx = randperm(numTrainImages,16);
figure
for i = 1:16
    subplot(4,4,i)
    I = readimage(imdsTrain,idx(i));
    imshow(I)
end
```



步骤 2：加载预训练好的网络

```
% 加载 alexnet 网络（注：该网络需要提前下载，当输入下面命令时按要求下载即可）
net = alexnet;
```

步骤 3：对网络结构进行改进

```
% 保留 AlexNet 倒数第三层之前的网络
layersTransfer = net.Layers(1:end-3);

% 确定训练数据中需要分类的种类
numClasses = numel(categories(imdsTrain.Labels));
```

```
% 构建新的网络，保留 AlexNet 倒数第三层之前的网络，在此之后重新添加了全连接
layers = [
    layersTransfer                                  % 保留 AlexNet 倒数第三层之前
的网络
    fullyConnectedLayer(numClasses)                 % 将新的全连接层的输出设置为
训练数据中的种类
    softmaxLayer                                    % 添加新的 Softmax 层
    classificationLayer ];                          % 添加新的分类层
```

步骤 4：调整数据集

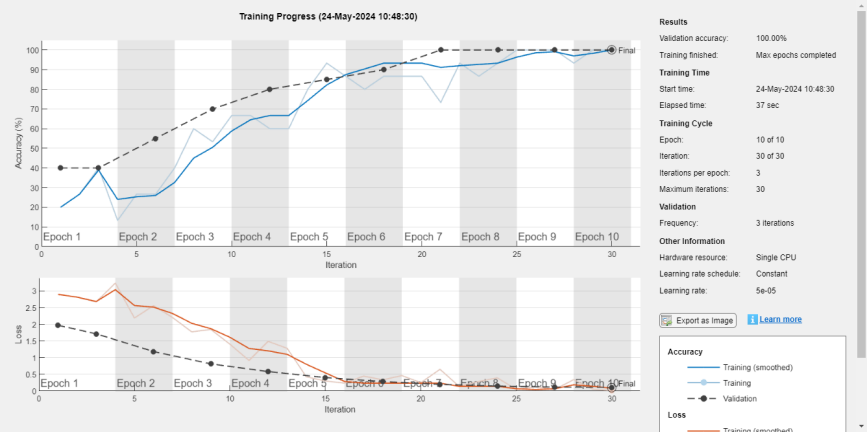
```
% 查看网络输入层的大小和通道数
inputSize = net.Layers(1).InputSize;

% 将训练图像的大小调整为与输入层的大小相同
augimdsTrain = augmentedImageDatastore(inputSize(1:2),imdsTrain);

% 将验证图像的大小调整为与输入层的大小相同
augimdsValidation = augmentedImageDatastore(inputSize(1:2),imdsValidation);
```

对网络进行训练

```
% 对训练参数进行设置
options = trainingOptions('sgdm', ...
    'MiniBatchSize',15, ...
    'MaxEpochs',10, ...
    'InitialLearnRate',0.00005, ...
    'Shuffle','every-epoch', ...
    'ValidationData',augimdsValidation, ...
    'ValidationFrequency',3, ...
    'Verbose',true, ...
    'Plots','training-progress');

% 用训练图像对网络进行训练
netTransfer = trainNetwork(augimdsTrain,layers,options);
```

```
Training on single CPU.
Initializing input data normalization.
|=========================================================================================================================
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Validation  |  Mini-batch  |  Validation  |  Base Learni
|         |             |   (hh:mm:ss)   |   Accuracy   |   Accuracy   |     Loss     |     Loss     |      Rate
|=========================================================================================================================
|       1 |           1 |       00:00:03 |       20.00% |       40.00% |       2.8960 |       1.9698 |      5.0000e
|       1 |           3 |       00:00:06 |       40.00% |       40.00% |       2.6730 |       1.7133 |      5.0000e
|       2 |           6 |       00:00:09 |       26.67% |       55.00% |       2.5593 |       1.1835 |      5.0000e
|       3 |           9 |       00:00:13 |       53.33% |       70.00% |       1.8450 |       0.8137 |      5.0000e
|       4 |          12 |       00:00:16 |       60.00% |       80.00% |       1.4889 |       0.5832 |      5.0000e
|       5 |          15 |       00:00:20 |       93.33% |       85.00% |       0.3012 |       0.4038 |      5.0000e
|       6 |          18 |       00:00:23 |       86.67% |       90.00% |       0.3359 |       0.2840 |      5.0000e
|       7 |          21 |       00:00:27 |       73.33% |      100.00% |       0.6506 |       0.1995 |      5.0000e
|       8 |          24 |       00:00:30 |       93.33% |      100.00% |       0.3955 |       0.1478 |      5.0000e
|       9 |          27 |       00:00:34 |      100.00% |      100.00% |       0.0428 |       0.1179 |      5.0000e
|      10 |          30 |       00:00:37 |      100.00% |      100.00% |       0.0134 |       0.0993 |      5.0000e
```

```
|=========================================================================================================================================|
Training finished: Max epochs completed.
```



分类验证图像并并随机显示分类结果

```
% 对训练好的网络采用验证数据集进行验证
[YPred,scores] = classify(netTransfer,augimdsValidation);

% 随机显示验证效果
idx = randperm(numel(imdsValidation.Files),4);
figure
for i = 1:4
    subplot(2,2,i)
    I = readimage(imdsValidation,idx(i));
    imshow(I)
    label = YPred(idx(i));
    title(string(label));
end
```

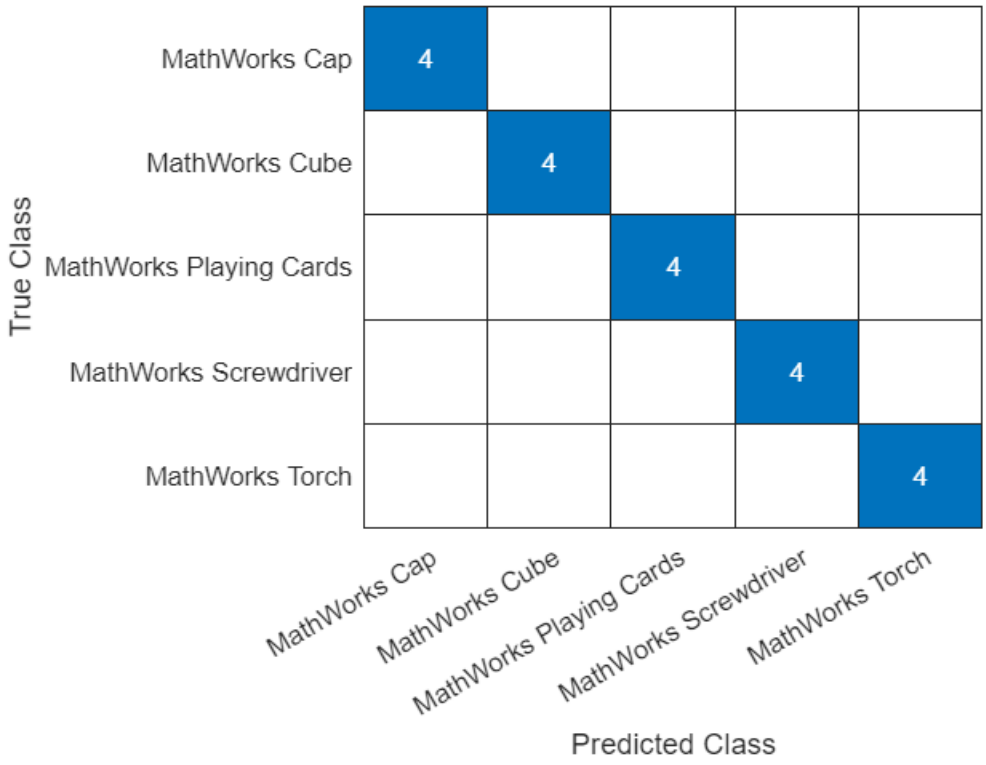
计算分类准确率

```
YValidation = imdsValidation.Labels;
accuracy = mean(YPred == YValidation)
```

```
accuracy = 1
```

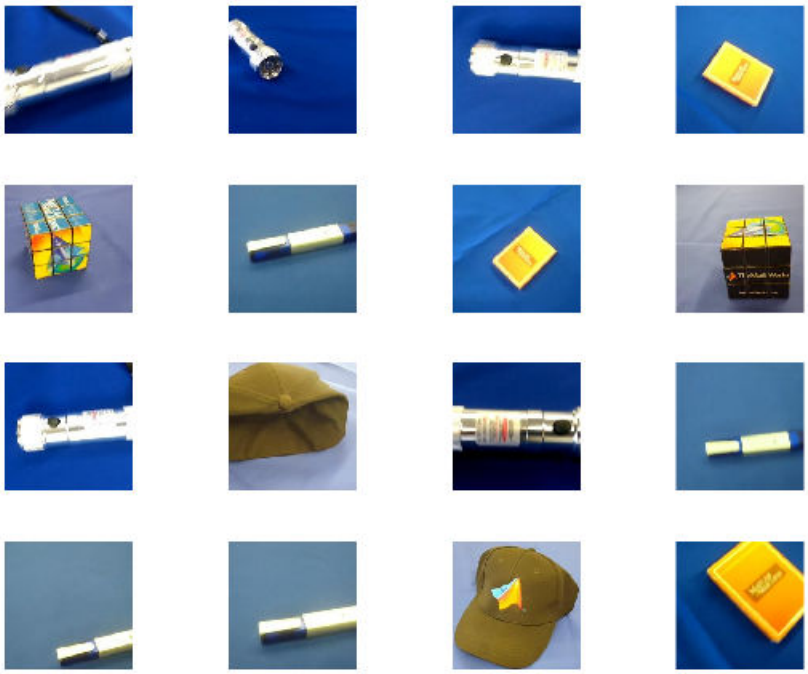
创建并显示混淆矩阵

```
figure
confusionchart(YValidation,YPred)
```



Exercise-5

```
% 加载图像数据
unzip('./data/MerchData.zip', './IMDS/MerchData');
imds = imageDatastore('IMDS/MerchData', ...
    'IncludeSubfolders',true, ...
    'LabelSource','foldernames');

% 划分验证集和训练集
[imdsTrain,imdsValidation] = splitEachLabel(imds,0.7,'randomized');

% 随机显示训练集中的部分图像
numTrainImages = numel(imdsTrain.Labels);
idx = randperm(numTrainImages,16);
figure
for i = 1:16
    subplot(4,4,i)
    I = readimage(imdsTrain,idx(i));
    imshow(I)
end
```



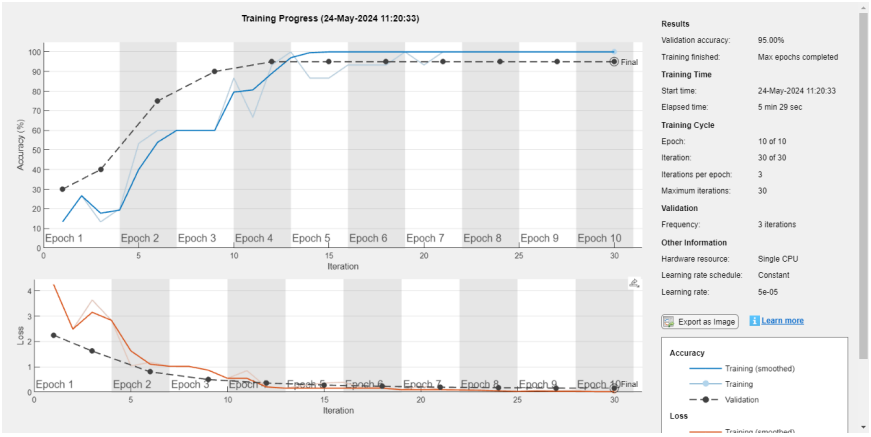
```
net = vgg16;
% analyzeNetwork(net)
```

```
layersTransfer = net.Layers(1:end-3);

numClasses = numel(categories(imdsTrain.Labels));

layers = [
    layersTransfer
    fullyConnectedLayer(numClasses)
    softmaxLayer
    classificationLayer
    ];
```

```
% 查看网络输入层的大小和通道数
inputSize = net.Layers(1).InputSize;

% 将训练图像的大小调整为与输入层的大小相同
augimdsTrain = augmentedImageDatastore(inputSize(1:2),imdsTrain);

% 将验证图像的大小调整为与输入层的大小相同
augimdsValidation = augmentedImageDatastore(inputSize(1:2),imdsValidation);
```

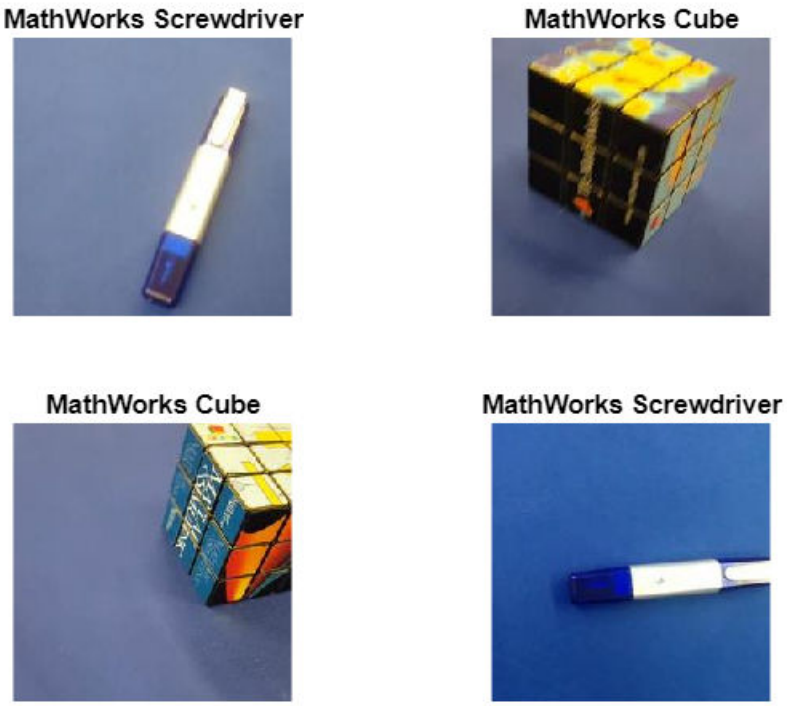
```
% 对训练参数进行设置
options = trainingOptions('sgdm', ...
    'MiniBatchSize',15, ...
    'MaxEpochs',10, ...
    'InitialLearnRate',0.00005, ...
    'Shuffle','every-epoch', ...
    'ValidationData',augimdsValidation, ...
    'ValidationFrequency',3, ...
    'Verbose',true, ...
    'Plots','training-progress');

% 用训练图像对网络进行训练
netTransfer = trainNetwork(augimdsTrain,layers,options);
```

```
Training on single CPU.
Initializing input data normalization.
|=========================================================================================================================================|
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Validation  |  Mini-batch  |  Validation  |  Base Learnin|
|         |             |   (hh:mm:ss)   |   Accuracy   |   Accuracy   |     Loss     |     Loss     |      Rate    |
|=========================================================================================================================================|
|       1 |           1 |       00:00:15 |       13.33% |       30.00% |       4.2538 |       2.2510 |       5.0000e-|
|       1 |           3 |       00:00:40 |       13.33% |       40.00% |       3.6386 |       1.6201 |       5.0000e-|
|       2 |           6 |       00:01:12 |       60.00% |       75.00% |       1.1786 |       0.7998 |       5.0000e-|
|       3 |           9 |       00:01:44 |       60.00% |       90.00% |       0.8684 |       0.4971 |       5.0000e-|
|       4 |          12 |       00:02:16 |       93.33% |       95.00% |       0.2067 |       0.3588 |       5.0000e-|
|       5 |          15 |       00:02:48 |       86.67% |       95.00% |       0.3604 |       0.2800 |       5.0000e-|
|       6 |          18 |       00:03:20 |       93.33% |       95.00% |       0.1945 |       0.2284 |       5.0000e-|
|       7 |          21 |       00:03:51 |      100.00% |       95.00% |       0.1051 |       0.1951 |       5.0000e-|
|       8 |          24 |       00:04:21 |      100.00% |       95.00% |       0.0502 |       0.1748 |       5.0000e-|
|       9 |          27 |       00:04:54 |      100.00% |       95.00% |       0.0585 |       0.1611 |       5.0000e-|
|      10 |          30 |       00:05:26 |      100.00% |       95.00% |       0.0267 |       0.1495 |       5.0000e-|
|=========================================================================================================================================|
Training finished: Max epochs completed.
```




```
% 对训练好的网络采用验证数据集进行验证
[YPred,scores] = classify(netTransfer,augimdsValidation);

% 随机显示验证效果
idx = randperm(numel(imdsValidation.Files),4);
figure
for i = 1:4
```

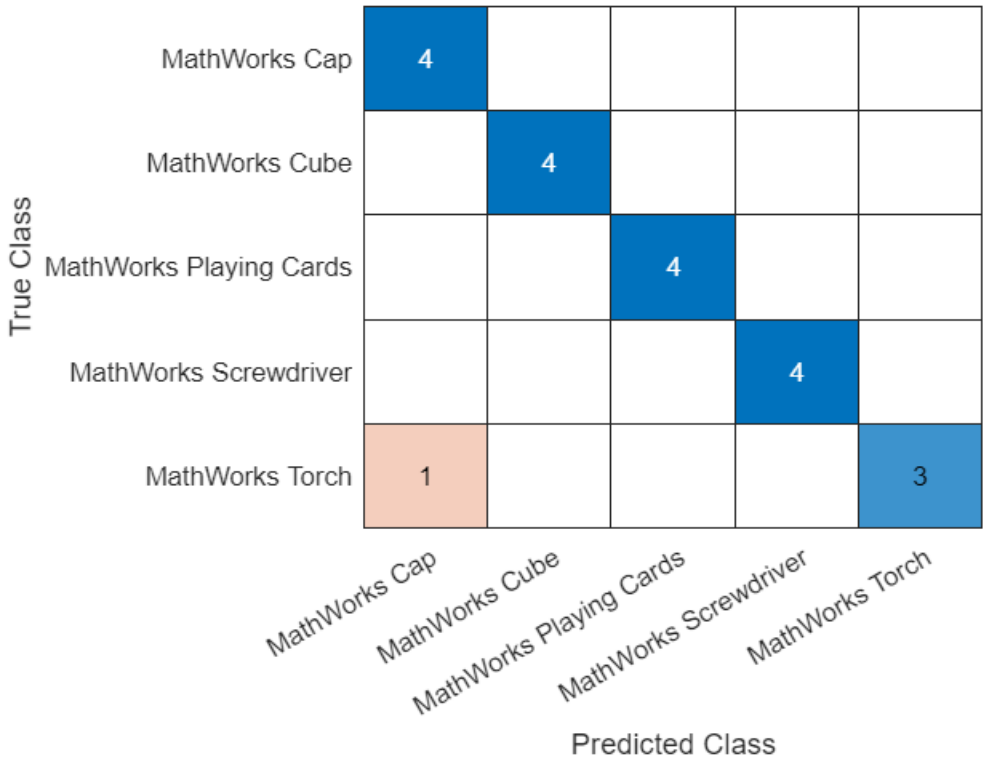
```
    subplot(2,2,i)
    I = readimage(imdsValidation,idx(i));
    imshow(I)
    label = YPred(idx(i));
    title(string(label));
end
```




```
YValidation = imdsValidation.Labels;
accuracy = mean(YPred == YValidation)
```

```
accuracy = 0.9500
```

```
figure
confusionchart(YValidation,YPred)
```



Exercise-6

```
% 加载图像数据
unzip('./data/small_EUROSat.zip', './IMDS/small_EUROSat');
imds = imageDatastore('IMDS/small_EUROSat', ...
    'IncludeSubfolders',true, ...
    'LabelSource','foldernames');

% 划分验证集和训练集
[imdsTrain,imdsValidation] = splitEachLabel(imds,0.7,'randomized');

% 随机显示训练集中的部分图像
numTrainImages = numel(imdsTrain.Labels);
idx = randperm(numTrainImages,16);
figure
for i = 1:16
    subplot(4,4,i)
    I = readimage(imdsTrain,idx(i));
    imshow(I)
end
```

```
net = alexnet;
```

```
layersTransfer = net.Layers(1:end-3);

numClasses = numel(categories(imdsTrain.Labels));

layers = [
    layersTransfer
    fullyConnectedLayer(numClasses)
    softmaxLayer
    classificationLayer
    ];
```

```
% 查看网络输入层的大小和通道数
inputSize = net.Layers(1).InputSize;

% 将训练图像的大小调整为与输入层的大小相同
augimdsTrain = augmentedImageDatastore(inputSize(1:2),imdsTrain);

% 将验证图像的大小调整为与输入层的大小相同
augimdsValidation = augmentedImageDatastore(inputSize(1:2),imdsValidation);
```

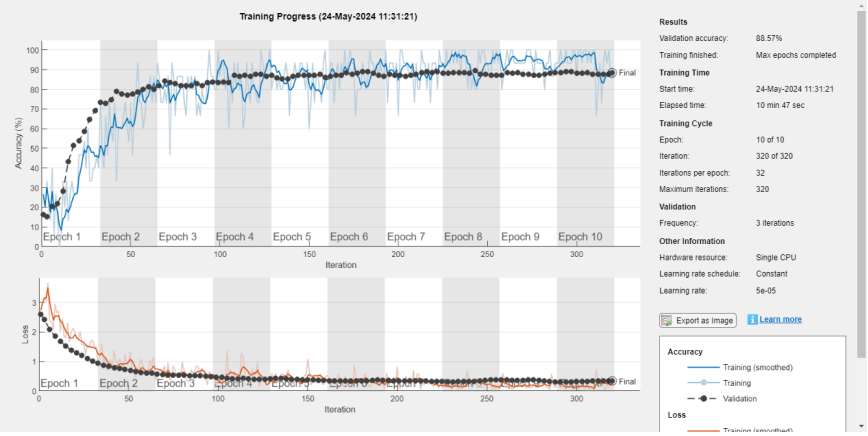
```
% 对训练参数进行设置
```

```
options = trainingOptions('sgdm', ...
    'MiniBatchSize',15, ...
    'MaxEpochs',10, ...
    'InitialLearnRate',0.00005, ...
    'Shuffle','every-epoch', ...
    'ValidationData',augimdsValidation, ...
    'ValidationFrequency',3, ...
    'Verbose',true, ...
    'Plots','training-progress');

% 用训练图像对网络进行训练
netTransfer = trainNetwork(augimdsTrain,layers,options);
```

```
Training on single CPU.
Initializing input data normalization.
|=========================================================================================================================
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Validation  |  Mini-batch  |  Validation  |  Base Learnin
|         |             |   (hh:mm:ss)   |   Accuracy   |   Accuracy   |     Loss     |     Loss     |     Rate
|=========================================================================================================================
|       1 |           1 |       00:00:05 |       26.67% |       16.19% |       2.7319 |       2.6052 |       5.0000e-
|       1 |           3 |       00:00:10 |       33.33% |       15.24% |       3.1596 |       2.4217 |       5.0000e-
|       1 |           6 |       00:00:17 |       40.00% |       20.48% |       2.3993 |       2.0914 |       5.0000e-
|       1 |           9 |       00:00:23 |       13.33% |       21.90% |       2.9472 |       1.8524 |       5.0000e-
|       1 |          12 |       00:00:29 |       33.33% |       28.10% |       2.0265 |       1.6816 |       5.0000e-
|       1 |          15 |       00:00:37 |       13.33% |       43.33% |       1.7954 |       1.5190 |       5.0000e-
|       1 |          18 |       00:00:44 |       26.67% |       51.43% |       2.0007 |       1.3903 |       5.0000e-
|       1 |          21 |       00:00:51 |       46.67% |       53.81% |       1.7703 |       1.2881 |       5.0000e-
|       1 |          24 |       00:00:58 |       53.33% |       58.57% |       1.5739 |       1.1988 |       5.0000e-
|       1 |          27 |       00:01:05 |       33.33% |       64.76% |       1.5568 |       1.1012 |       5.0000e-
|       1 |          30 |       00:01:12 |       40.00% |       69.05% |       1.7096 |       1.0153 |       5.0000e-
|       2 |          33 |       00:01:18 |       73.33% |       73.33% |       0.8832 |       0.9422 |       5.0000e-
|       2 |          36 |       00:01:25 |       66.67% |       72.86% |       0.9891 |       0.8790 |       5.0000e-
|       2 |          39 |       00:01:31 |       60.00% |       74.76% |       0.7942 |       0.8307 |       5.0000e-
|       2 |          42 |       00:01:38 |       33.33% |       79.05% |       1.3617 |       0.7865 |       5.0000e-
|       2 |          45 |       00:01:44 |       60.00% |       77.62% |       1.0756 |       0.7593 |       5.0000e-
|       2 |          48 |       00:01:50 |       60.00% |       77.14% |       1.2447 |       0.7337 |       5.0000e-
|       2 |          50 |       00:01:52 |       73.33% |              |       0.9325 |              |       5.0000e-
|       2 |          51 |       00:01:56 |       46.67% |       77.62% |       1.3321 |       0.6999 |       5.0000e-
|       2 |          54 |       00:02:02 |       93.33% |       78.57% |       0.4563 |       0.6647 |       5.0000e-
|       2 |          57 |       00:02:09 |       60.00% |       80.00% |       1.0804 |       0.6333 |       5.0000e-
|       2 |          60 |       00:02:15 |       80.00% |       81.43% |       0.5900 |       0.6132 |       5.0000e-
|       2 |          63 |       00:02:21 |       86.67% |       80.00% |       0.8373 |       0.6085 |       5.0000e-
|       3 |          66 |       00:02:27 |       86.67% |       81.90% |       0.4571 |       0.5805 |       5.0000e-
|       3 |          69 |       00:02:33 |       80.00% |       84.29% |       0.6447 |       0.5528 |       5.0000e-
|       3 |          72 |       00:02:39 |       46.67% |       83.33% |       0.9593 |       0.5435 |       5.0000e-
|       3 |          75 |       00:02:44 |       66.67% |       82.86% |       0.8515 |       0.5435 |       5.0000e-
|       3 |          78 |       00:02:50 |       86.67% |       81.90% |       0.4513 |       0.5433 |       5.0000e-
|       3 |          81 |       00:02:56 |       66.67% |       81.90% |       0.6999 |       0.5311 |       5.0000e-
|       3 |          84 |       00:03:02 |       86.67% |       81.90% |       0.3698 |       0.5266 |       5.0000e-
|       3 |          87 |       00:03:08 |       66.67% |       82.86% |       0.7184 |       0.5181 |       5.0000e-
|       3 |          90 |       00:03:14 |       80.00% |       81.90% |       0.5313 |       0.5202 |       5.0000e-
|       3 |          93 |       00:03:19 |       86.67% |       83.33% |       0.4078 |       0.5051 |       5.0000e-
|       3 |          96 |       00:03:25 |       86.67% |       83.81% |       0.5238 |       0.4862 |       5.0000e-
|       4 |          99 |       00:03:31 |      100.00% |       83.33% |       0.1783 |       0.4696 |       5.0000e-
|       4 |         100 |       00:03:32 |       86.67% |              |       0.4283 |              |       5.0000e-
|       4 |         102 |       00:03:37 |       93.33% |       83.81% |       0.1986 |       0.4609 |       5.0000e-
|       4 |         105 |       00:03:43 |       60.00% |       83.33% |       1.3591 |       0.4376 |       5.0000e-
|       4 |         108 |       00:03:48 |       93.33% |       87.14% |       0.4268 |       0.4119 |       5.0000e-
|       4 |         111 |       00:03:54 |       80.00% |       86.67% |       0.4257 |       0.4167 |       5.0000e-
|       4 |         114 |       00:04:00 |       66.67% |       87.14% |       1.1513 |       0.4280 |       5.0000e-
|       4 |         117 |       00:04:06 |       73.33% |       86.67% |       0.5201 |       0.4120 |       5.0000e-
```

4	120	00:04:11	80.00%	87.62%	0.5337	0.3955	5.0000e-
4	123	00:04:17	100.00%	87.62%	0.1491	0.3894	5.0000e-
4	126	00:04:23	93.33%	86.67%	0.3021	0.4010	5.0000e-
5	129	00:04:29	80.00%	87.14%	0.4319	0.4159	5.0000e-
5	132	00:04:35	80.00%	85.71%	0.5911	0.4231	5.0000e-
5	135	00:04:41	80.00%	85.24%	0.5734	0.4249	5.0000e-
5	138	00:04:47	86.67%	85.24%	0.4548	0.4148	5.0000e-
5	141	00:04:52	86.67%	86.67%	0.2876	0.4045	5.0000e-
5	144	00:04:58	86.67%	87.14%	0.3780	0.3912	5.0000e-
5	147	00:05:04	86.67%	87.14%	0.3300	0.3805	5.0000e-
5	150	00:05:10	60.00%	87.14%	1.0725	0.3821	5.0000e-
5	153	00:05:15	73.33%	87.14%	0.4163	0.3790	5.0000e-
5	156	00:05:21	80.00%	87.62%	0.4976	0.3692	5.0000e-
5	159	00:05:27	100.00%	86.19%	0.2339	0.3616	5.0000e-
6	162	00:05:32	93.33%	86.67%	0.4212	0.3509	5.0000e-
6	165	00:05:38	86.67%	87.14%	0.3975	0.3422	5.0000e-
6	168	00:05:44	93.33%	87.14%	0.4199	0.3422	5.0000e-
6	171	00:05:50	86.67%	88.57%	0.5182	0.3409	5.0000e-
6	174	00:05:55	93.33%	87.62%	0.3262	0.3387	5.0000e-
6	177	00:06:01	80.00%	88.10%	0.4136	0.3354	5.0000e-
6	180	00:06:07	93.33%	89.05%	0.2599	0.3410	5.0000e-
6	183	00:06:12	86.67%	89.05%	0.2909	0.3514	5.0000e-
6	186	00:06:18	93.33%	88.10%	0.3306	0.3621	5.0000e-
6	189	00:06:24	86.67%	87.14%	0.2625	0.3709	5.0000e-
6	192	00:06:30	86.67%	86.67%	0.2141	0.3663	5.0000e-
7	195	00:06:35	93.33%	87.62%	0.1671	0.3543	5.0000e-
7	198	00:06:41	86.67%	87.14%	0.4691	0.3443	5.0000e-
7	200	00:06:43	66.67%		0.8051		5.0000e-
7	201	00:06:47	100.00%	87.14%	0.0862	0.3407	5.0000e-
7	204	00:06:53	93.33%	86.67%	0.4144	0.3406	5.0000e-
7	207	00:06:58	86.67%	87.14%	0.4658	0.3454	5.0000e-
7	210	00:07:04	93.33%	87.62%	0.1694	0.3393	5.0000e-
7	213	00:07:10	100.00%	88.57%	0.0656	0.3368	5.0000e-
7	216	00:07:16	80.00%	89.05%	0.4125	0.3370	5.0000e-
7	219	00:07:21	93.33%	88.57%	0.2527	0.3315	5.0000e-
7	222	00:07:28	80.00%	89.05%	0.4015	0.3253	5.0000e-
8	225	00:07:34	93.33%	88.10%	0.2045	0.3181	5.0000e-
8	228	00:07:40	93.33%	88.10%	0.1840	0.3138	5.0000e-
8	231	00:07:46	93.33%	88.57%	0.2417	0.3168	5.0000e-
8	234	00:07:52	100.00%	88.57%	0.0306	0.3247	5.0000e-
8	237	00:07:58	100.00%	88.57%	0.1797	0.3312	5.0000e-
8	240	00:08:03	93.33%	88.10%	0.3140	0.3334	5.0000e-
8	243	00:08:09	80.00%	89.52%	0.4092	0.3327	5.0000e-
8	246	00:08:15	93.33%	87.62%	0.2107	0.3408	5.0000e-
8	249	00:08:21	93.33%	87.62%	0.2633	0.3614	5.0000e-
8	250	00:08:22	93.33%		0.2107		5.0000e-
8	252	00:08:27	86.67%	87.14%	0.3687	0.3770	5.0000e-
8	255	00:08:33	100.00%	87.14%	0.0846	0.3801	5.0000e-
9	258	00:08:39	80.00%	87.14%	0.4305	0.3786	5.0000e-
9	261	00:08:45	100.00%	88.57%	0.0965	0.3723	5.0000e-
9	264	00:08:51	100.00%	88.10%	0.0610	0.3650	5.0000e-
9	267	00:08:57	93.33%	88.57%	0.2017	0.3658	5.0000e-
9	270	00:09:03	93.33%	87.62%	0.2949	0.3678	5.0000e-
9	273	00:09:09	100.00%	87.62%	0.1271	0.3827	5.0000e-
9	276	00:09:15	86.67%	87.14%	0.2404	0.3726	5.0000e-
9	279	00:09:21	93.33%	87.14%	0.1530	0.3560	5.0000e-
9	282	00:09:27	80.00%	87.62%	0.4004	0.3381	5.0000e-
9	285	00:09:33	100.00%	88.10%	0.0566	0.3242	5.0000e-
9	288	00:09:39	93.33%	88.57%	0.1172	0.3129	5.0000e-
10	291	00:09:45	100.00%	88.57%	0.1153	0.3097	5.0000e-
10	294	00:09:51	100.00%	89.05%	0.0447	0.3100	5.0000e-
10	297	00:09:57	93.33%	89.05%	0.2194	0.3157	5.0000e-
10	300	00:10:03	86.67%	88.10%	0.3722	0.3219	5.0000e-
10	303	00:10:09	93.33%	88.10%	0.2561	0.3260	5.0000e-

```
|      10 |         306 |       00:10:15 |       93.33% |       88.57% |       0.2372 |       0.3313 |       5.0000e-
|      10 |         309 |       00:10:21 |      100.00% |       87.62% |       0.0916 |       0.3352 |       5.0000e-
|      10 |         312 |       00:10:27 |       80.00% |       87.62% |       0.3500 |       0.3406 |       5.0000e-
|      10 |         315 |       00:10:33 |       93.33% |       87.62% |       0.1596 |       0.3398 |       5.0000e-
|      10 |         318 |       00:10:39 |      100.00% |       87.62% |       0.1212 |       0.3424 |       5.0000e-
|      10 |         320 |       00:10:44 |       86.67% |       88.57% |       0.2824 |       0.3358 |       5.0000e-
|=========================================================================================================================
Training finished: Max epochs completed.
```

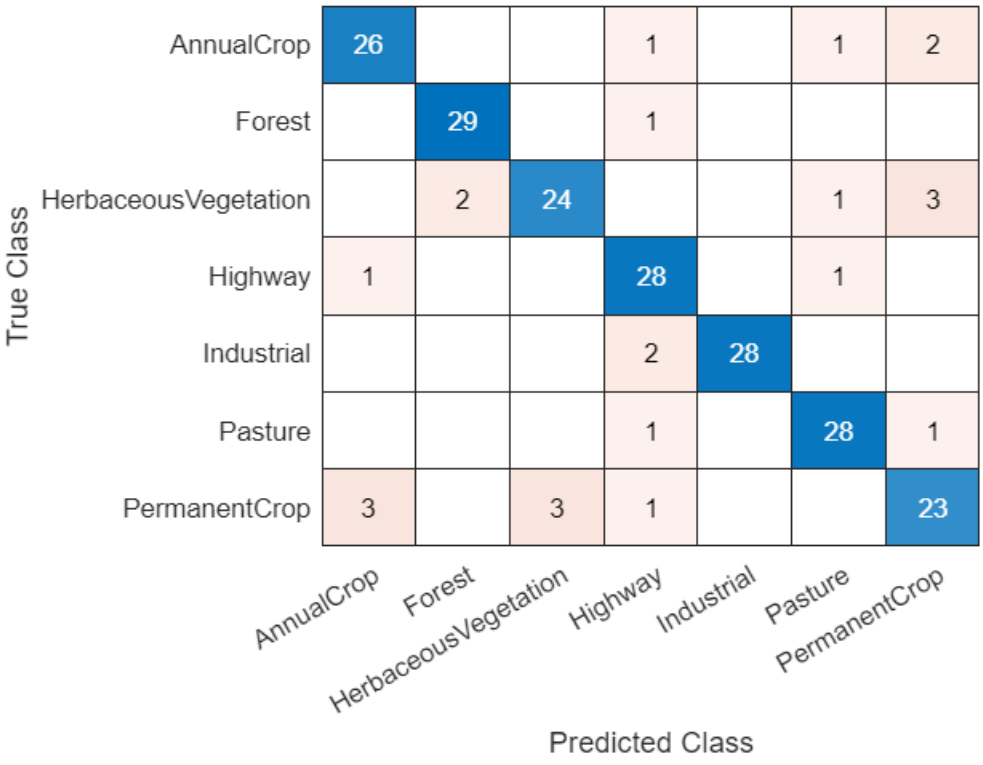



```
% 对训练好的网络采用验证数据集进行验证
[YPred,scores] = classify(netTransfer,augimdsValidation);

% 随机显示验证效果
idx = randperm(numel(imdsValidation.Files),4);
figure
for i = 1:4
    subplot(2,2,i)
    I = readimage(imdsValidation,idx(i));
    imshow(I)
    label = YPred(idx(i));
    title(string(label));
end
```


```
YValidation = imdsValidation.Labels;
accuracy = mean(YPred == YValidation)
```

```
accuracy = 0.8857
```

```
figure
confusionchart(YValidation,YPred)
```


AnnualCrop
Forest
HerbaceousVegetation
Highway
Industrial
Pasture
PermanentCrop
True Class
26 1 1 2
29 1
2 24 1 3
1 28 1
2 28
1 28 1
3 3 1 23
AnnualCrop
Forest
HerbaceousVegetation
Highway
Industrial
Pasture
PermanentCrop
Predicted Class