M231 - Pattern Recognition and Machine Learning

DUCK-HA HWANG
UID#: 404589112

Project 0: Experiments on Deep Learning with Cov. Neural Networks

There are 3 steps for you to do in this project. We will also ask you to run this code again in later projects for comparison.

We will host a Tutorial Session on Tuesday evening.

1. Download CIFAR-10, and learn a LeNet as described in Table 2 on it. Plot the training error and testing error against the training iteration.

Problem (1): Answer

2. Keep the Block5, learn a ConvNet only with : a) Block1; b) Block1 and Block2; c) Block1, Block2 and Block3 respectively. Compare the _nal training errors and testing errors with LeNet in a table. (Hint: You need to change the _lter size in Block5 to match the output from previous block.)

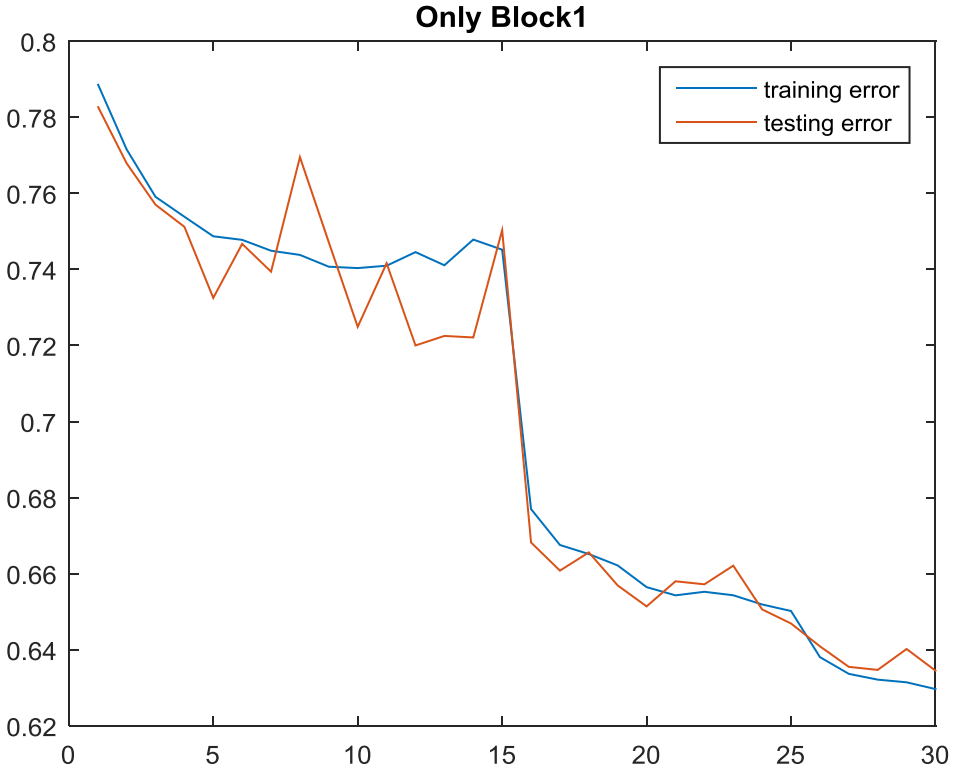
Problem2-(a) Answer

```
% Block 1
net.layers{end+1} = struct('type', 'conv', ...
                           'weights', {{0.01*randn(5,5,3,32, 'single'), zeros(1, 32, 'single')}}, ...
                           'learningRate', lr, ...
                           'stride', 1, ...
                           'pad', 2) ;
net.layers{end+1} = struct('type', 'pool', ...
                           'method', 'max', ...
                           'pool', [3 3], ...
                           'stride', 2, ...
                           'pad', [0 1 0 1]) ;
net.layers{end+1} = struct('type', 'relu') ;

% Block 5
net.layers{end+1} = struct('type', 'conv', ...
                           'weights', {{0.05*randn(16,16,32,10, 'single'), zeros(1,10,'single')}}, ...
                           'learningRate', .1*lr, ...
                           'stride', 1, ...
                           'pad', 0) ;
```

Changed opts.train.learningRate -> (opts.train.learningRate)*1/100

opts.train.learningRate = [0.05*ones(1,15) 0.005*ones(1,10) 0.0005*ones(1,5)] ;
opts.train.learningRate = [0.0005*ones(1,15) 0.00005*ones(1,10) 0.000005*ones(1,5)] ;

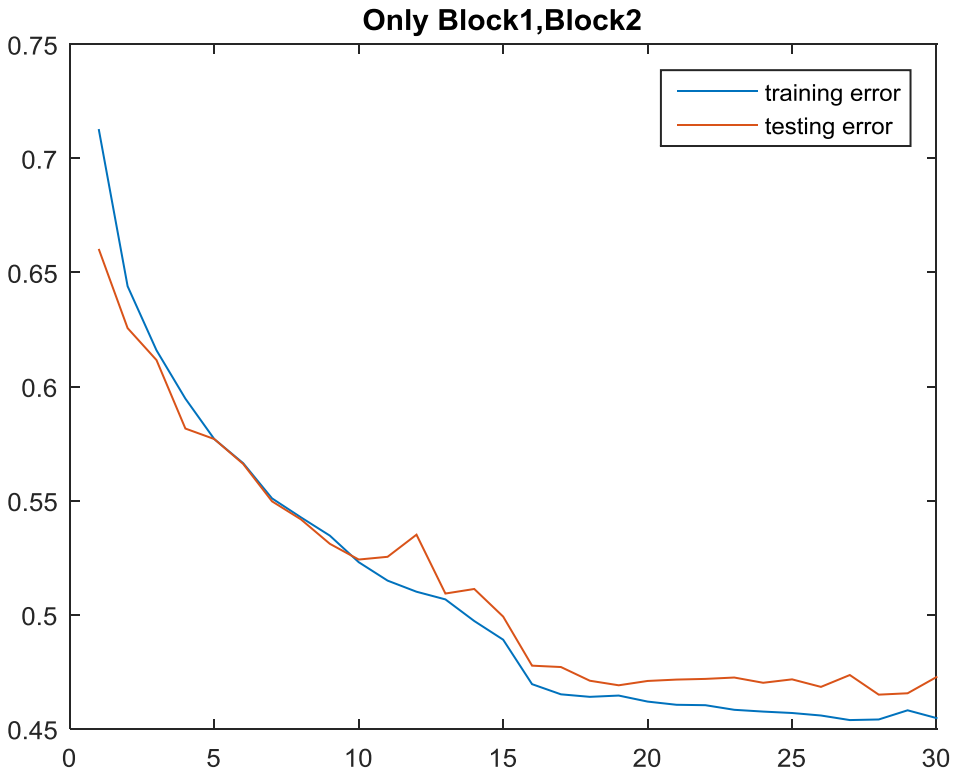
It will allow the model to train slowly, as there are too many nodes as an input to final connected layer, convergence is difficult to achieve. Reducing leaning rate will increase the chances of attaining convergence if exists.



Problem2-(b) Answer

```
% Block 1
net.layers{end+1} = struct('type', 'conv', ...
                           'weights', {{0.01*randn(5,5,3,32, 'single'), zeros(1, 32, 'single')}}, ...
                           'learningRate', lr, ...
                           'stride', 1, ...
                           'pad', 2) ;
net.layers{end+1} = struct('type', 'pool', ...
                           'method', 'max', ...
                           'pool', [3 3], ...
                           'stride', 2, ...
                           'pad', [0 1 0 1]) ;
net.layers{end+1} = struct('type', 'relu') ;

% Block 2
net.layers{end+1} = struct('type', 'conv', ...
                           'weights', {{0.05*randn(5,5,32,32, 'single'), zeros(1,32,'single')}}, ...
                           'learningRate', lr, ...
                           'stride', 1, ...
                           'pad', 2) ;
net.layers{end+1} = struct('type', 'relu') ;
net.layers{end+1} = struct('type', 'pool', ...
                           'method', 'avg', ...
                           'pool', [3 3], ...
                           'stride', 2, ...
                           'pad', [0 1 0 1]) ; % Emulate caffe
```

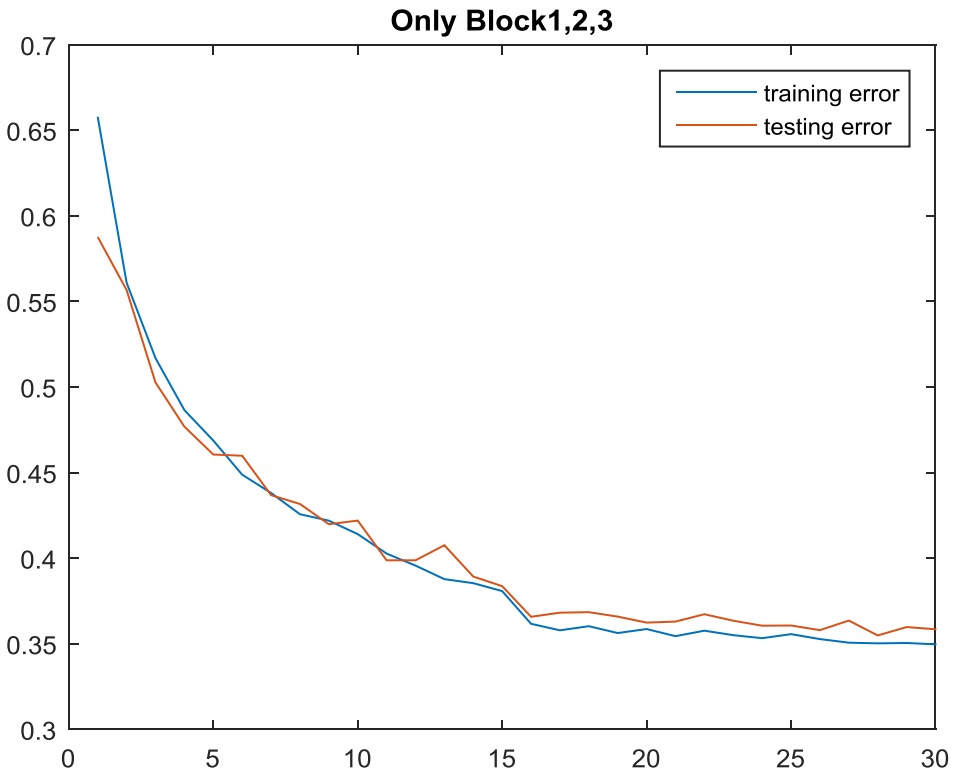
```
% Block 5
net.layers{end+1} = struct('type', 'conv', ...
                           'weights', {{0.05*randn(8,8,32,10, 'single'), zeros(1,10,'single')}}, ...
                           'learningRate', .1*lr, ...
                           'stride', 1, ...
                           'pad', 0) ;
```



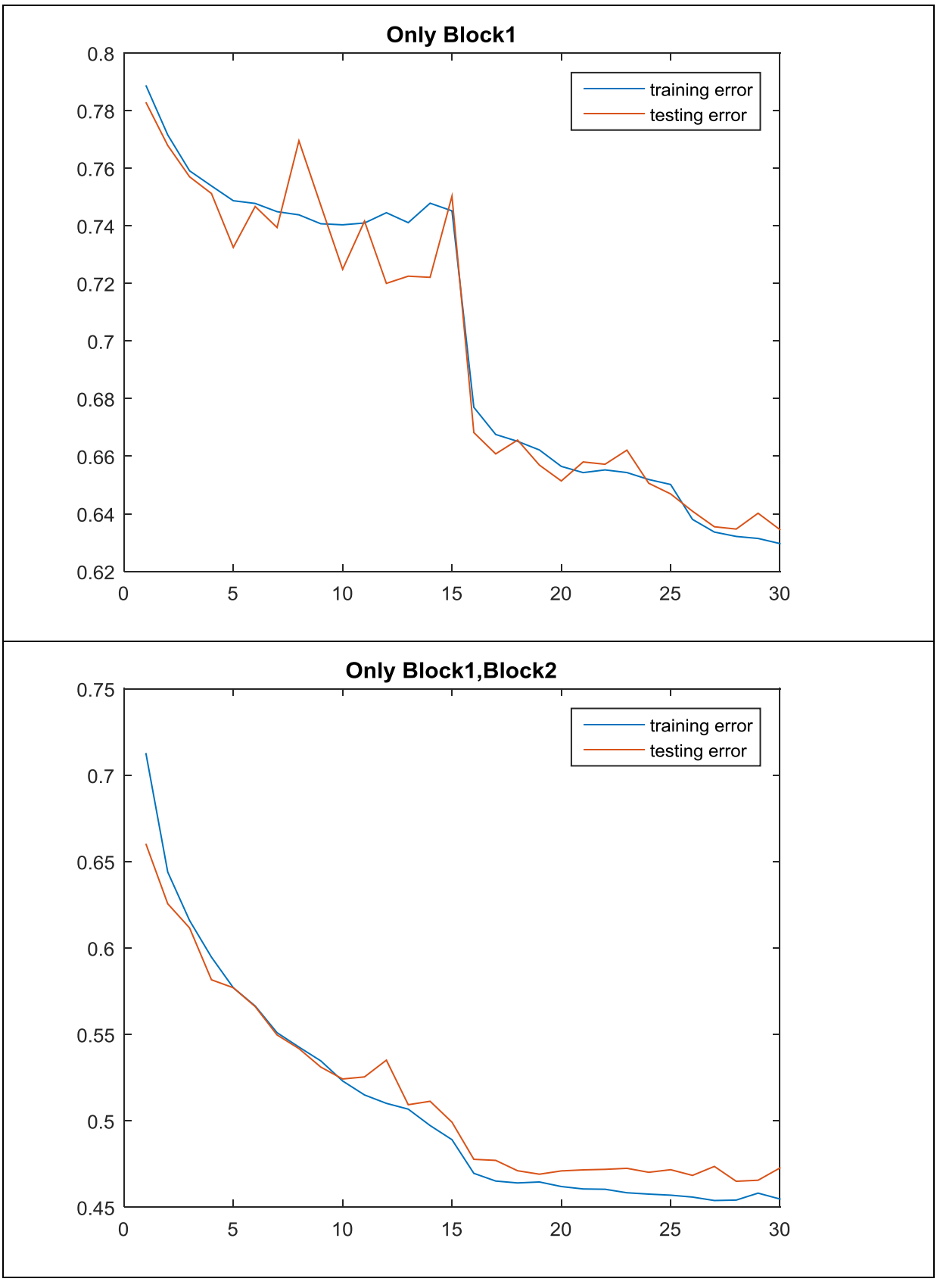
Problem2-(c) Answer

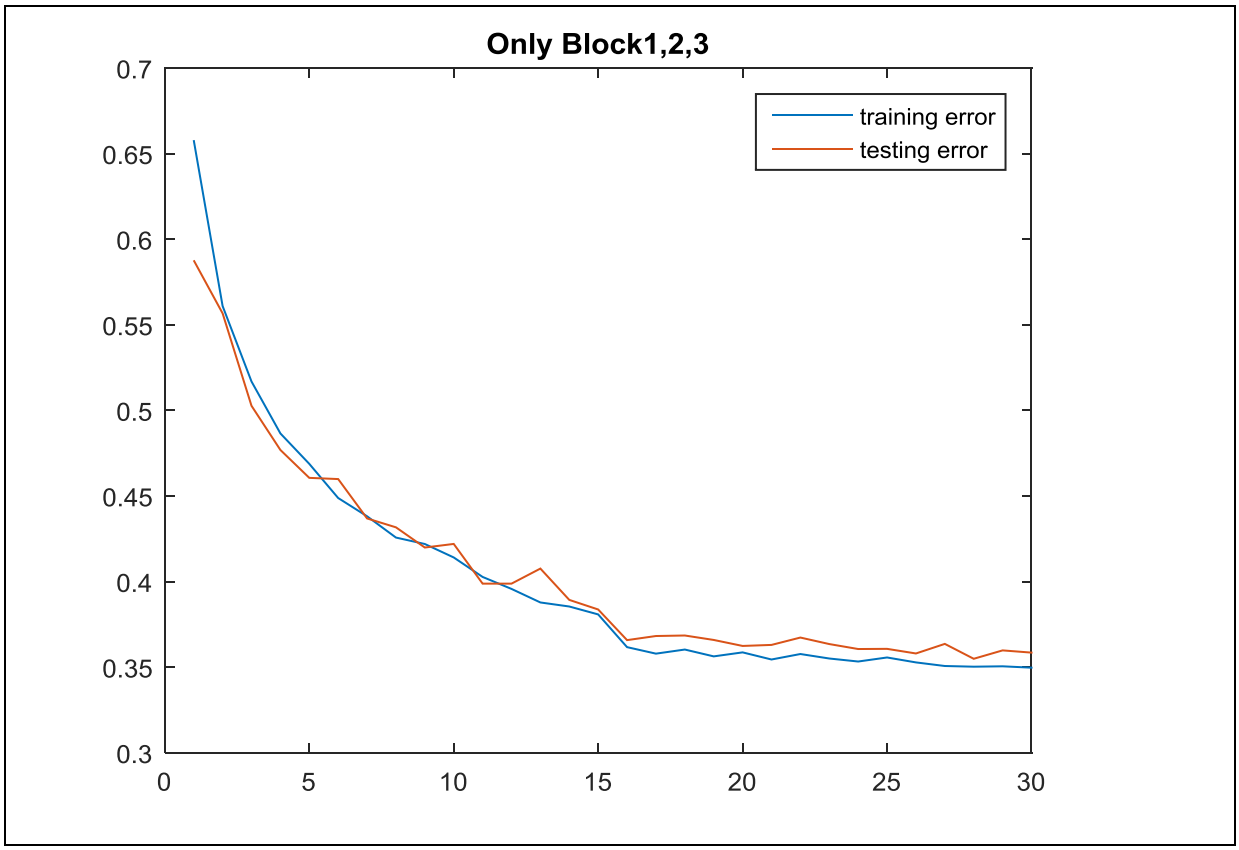
```
% Block 1
net.layers{end+1} = struct('type', 'conv', ...
                           'weights', {{0.01*randn(5,5,3,32, 'single'), zeros(1, 32, 'single')}}, ...
                           'learningRate', lr, ...
                           'stride', 1, ...
                           'pad', 2) ;
net.layers{end+1} = struct('type', 'pool', ...
                           'method', 'max', ...
                           'pool', [3 3], ...
                           'stride', 2, ...
                           'pad', [0 1 0 1]) ;
net.layers{end+1} = struct('type', 'relu') ;

% Block 2
net.layers{end+1} = struct('type', 'conv', ...
                           'weights', {{0.05*randn(5,5,32,32, 'single'), zeros(1,32,'single')}}, ...
                           'learningRate', lr, ...
                           'stride', 1, ...
                           'pad', 2) ;
net.layers{end+1} = struct('type', 'relu') ;
net.layers{end+1} = struct('type', 'pool', ...
                           'method', 'avg', ...
                           'pool', [3 3], ...
                           'stride', 2, ...
                           'pad', [0 1 0 1]) ; % Emulate caffe
```

```
% Block 3
net.layers{end+1} = struct('type', 'conv', ...
                           'weights', {{0.05*randn(5,5,32,64, 'single'), zeros(1,64,'single')}}, ...
                           'learningRate', lr, ...
                           'stride', 1, ...
                           'pad', 2) ;
net.layers{end+1} = struct('type', 'relu') ;
net.layers{end+1} = struct('type', 'pool', ...
                           'method', 'avg', ...
                           'pool', [3 3], ...
                           'stride', 2, ...
                           'pad', [0 1 0 1]) ; % Emulate caffe

% % Block 4
% net.layers{end+1} = struct('type', 'conv', ...
%                            'weights', {{0.05*randn(4,4,64,64, 'single'), zeros(1,64,'single')}}, ..
%                            'learningRate', lr, ...
%                            'stride', 1, ...
%                            'pad', 0) ;
% net.layers{end+1} = struct('type', 'relu') ;

% Block 5
net.layers{end+1} = struct('type', 'conv', ...
                           'weights', {{0.05*randn(4,4,64,10, 'single'), zeros(1,10,'single')}}, ...
                           'learningRate', .1*lr, ...
                           'stride', 1, ...
                           'pad', 0) ;
```


Only Block1,2,3
training error
testing error
0.7
0.65
0.6
0.55
0.5
0.45
0.4
0.35
0.3
0
5
10
15
20
25
30

Problem Compare case (a), (b), (c)





Analysis:

Using every Block (1, 2, 3, 4, 5) show the best result. (Converges to 0.2 error rate)

(a) Case converge to 0.63 error rate

(b) Case converge to 0.45 error rate

(c) Case converges to 0.35 error rate.

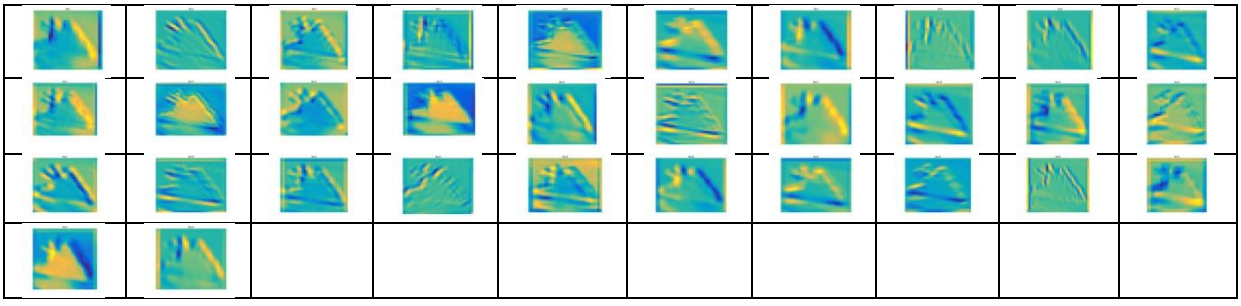
3. Filter response visualization. Figure 4 displays the visualization of filter responses computed by 32 filters in the first layer. The first color image is the input image, and the rest of 32 heat images are the corresponding filter response maps. The directory of ./code/examples/images/ contains 10 images from each category. For each image, visualize the filter response maps of 32 filters in the first layer. (Hint: For each image, you need to preprocess it by subtracting the mean image of CIFAR-10. The mean image is saved in net.averageImage. You can use res = vl_simplenn (net, image) to compute filter responses, and all filter responses computed by the first layer filters are saved in res(2).x)

Problem3-Answer

```
img1=imread('C:₩Users₩1234₩Desktop₩patternreconization₩code₩examples₩images₩1.png');

net=ans
net.layers= net.layers(1:end-1);
img = single(img1) - net.averageImage;
res = vl_simplenn(net, img);
figure, imagesc(res(2).x(:,:,1))

figure,imshow(img1)

for i=1:32
figure,imagesc(res(2).x(:,:,i));
axis off
title(['filter',num2str(i)])
saveas(gcf,['fileter ',num2str(i),'.jpg'])

end
```

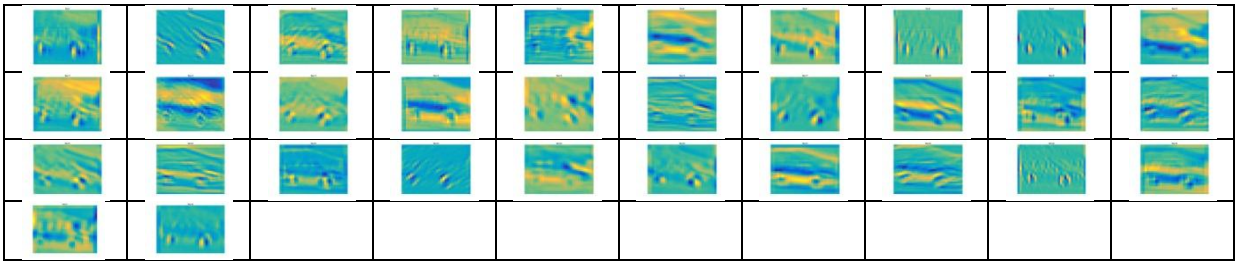
Original image_1

Each filter:



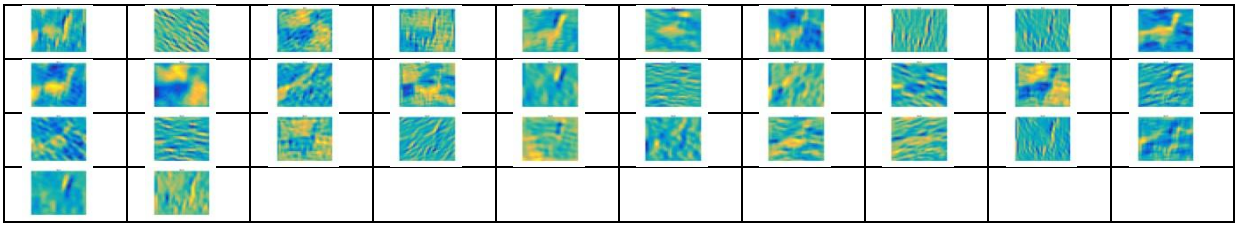
Original image_2

Each filter:



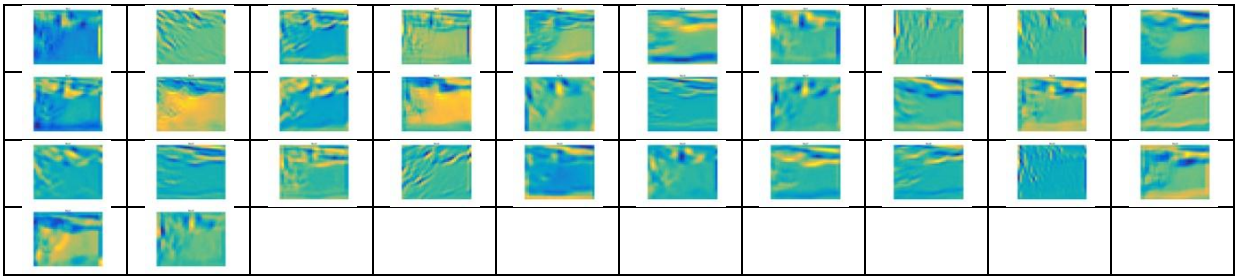
Original image_3

Each filter:



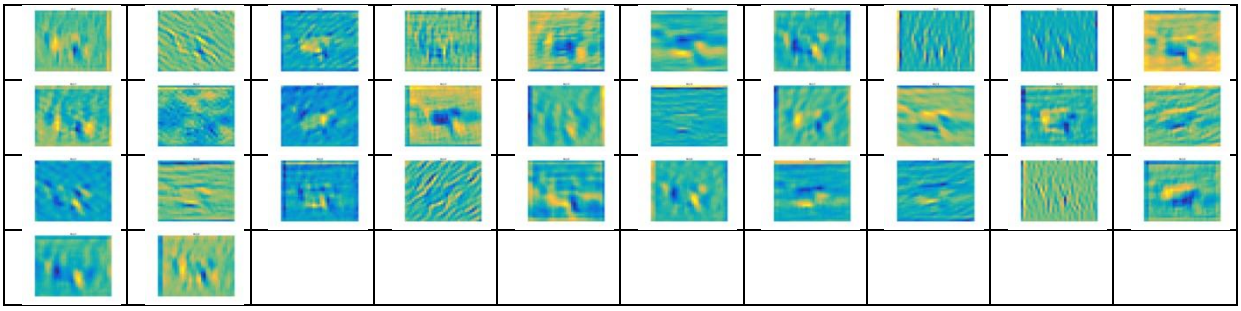
Original image_4

Each filter:



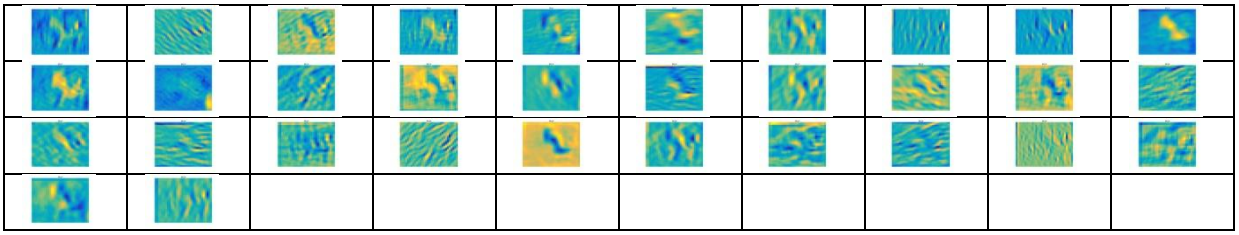
Original image_5

Each filter:

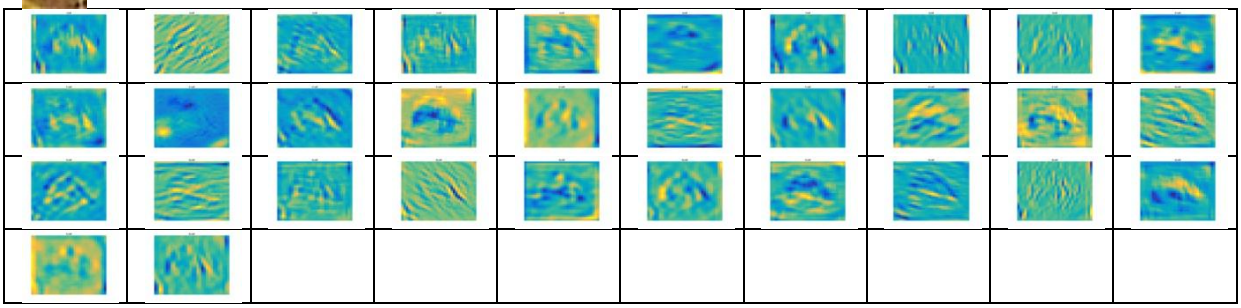


Original image_6

Each filter:

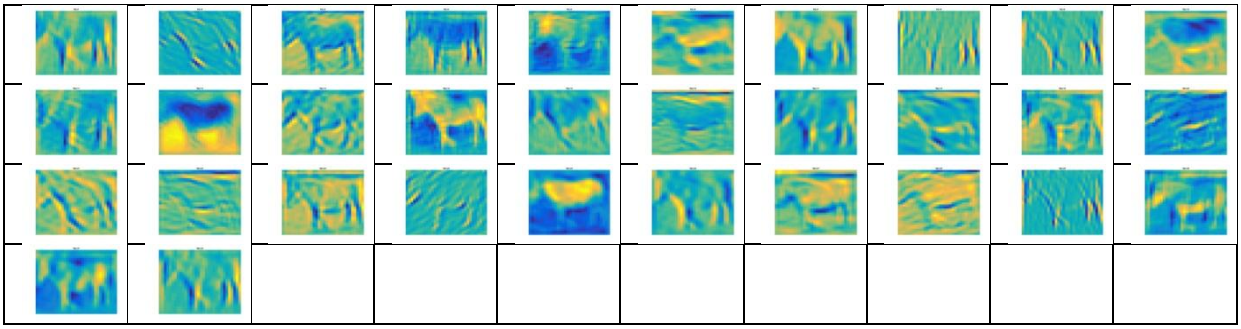


Original image_7



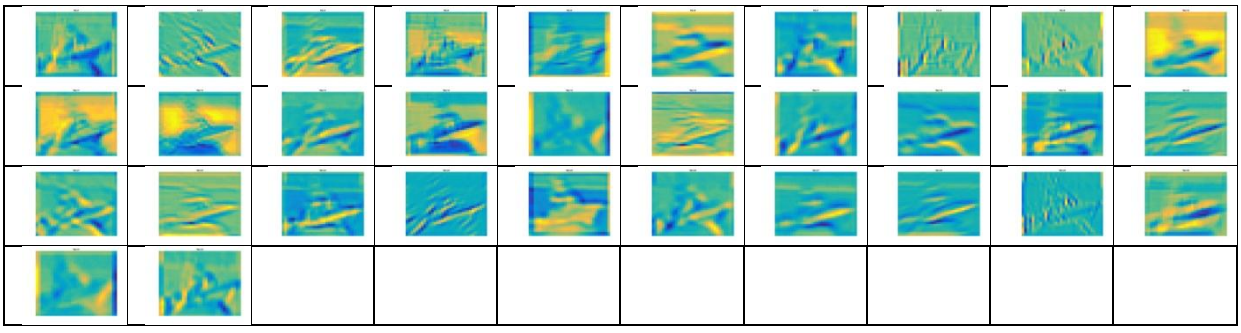
Original image_8

Each filter:



Original image_9

Each filter:



Original image_10

Each filter: