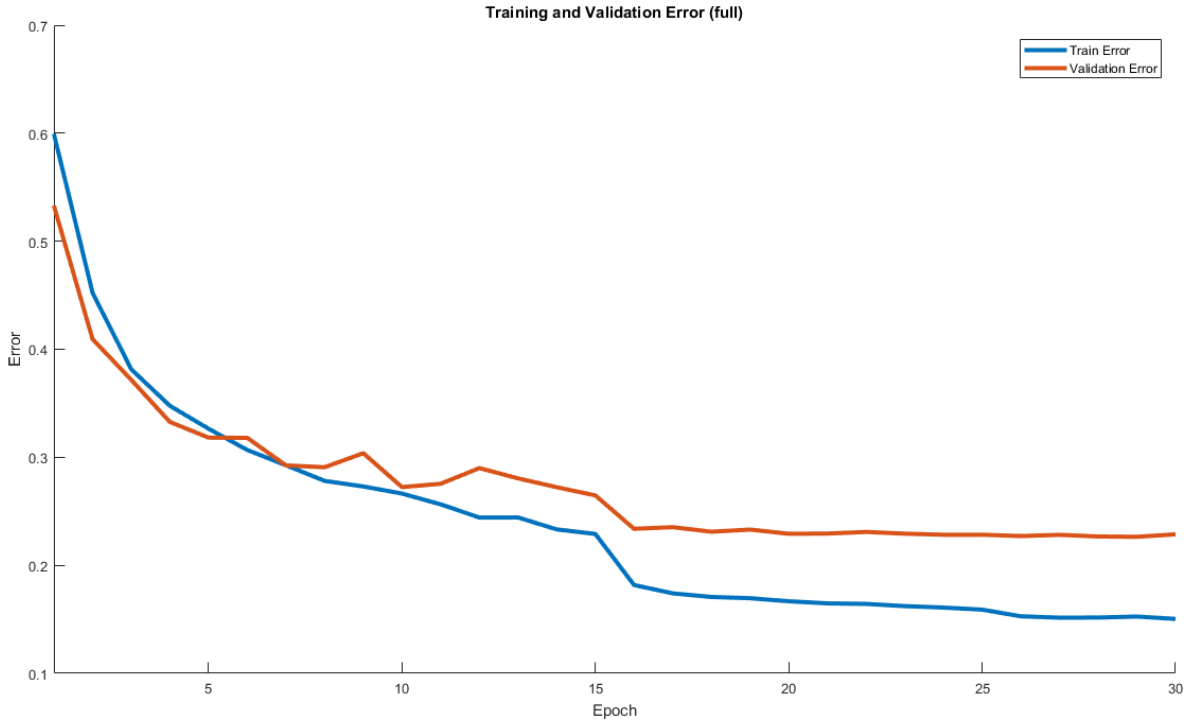
STATS 231A Project 0

Yufei Hu (404944367)

October 10, 2017

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.



Observations:

(1) As shown in the figure above, the validation error is often higher than training error under most cases. This is probably because the network is training using training dataset, but their weights do not get updated based on validation dataset. Therefore, these validation data are totally strange to the network while it has already 'seen' some training data during training. So it is reasonable to have observed this phenomenon.

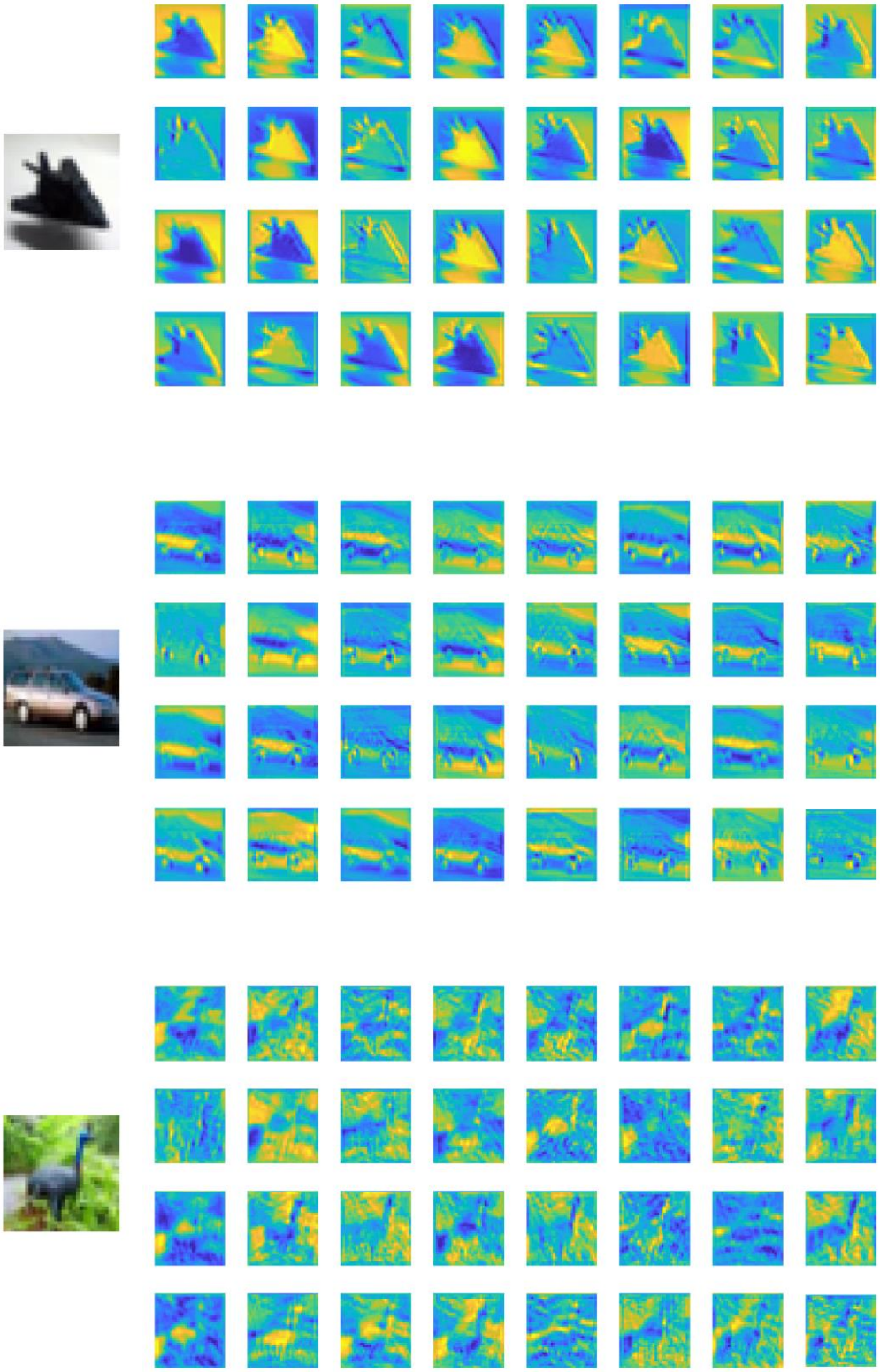
(2) Both training and validation errors decrease when the network goes through more epochs. This reason is quite straightforward as the network will have higher classification accuracy when the weights get more updates, meaning that these weights are moving from total random numbers towards a particulate pattern which better captures image features.

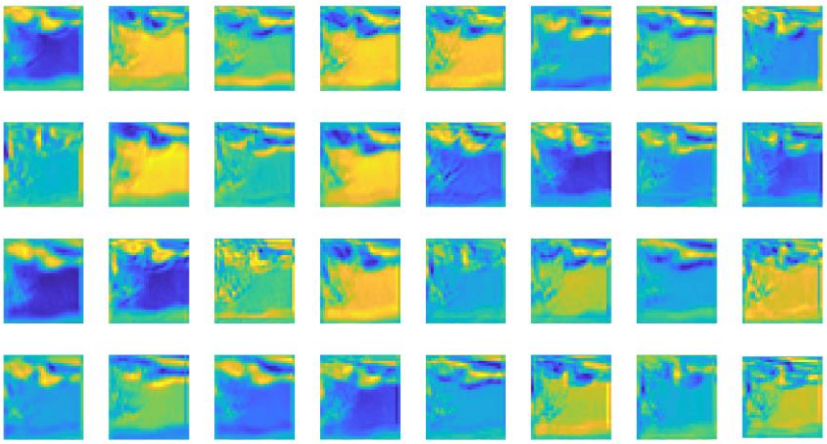
2. Keep the Block5, learn a ConvNet only with : a) Block1; b) Block1 and Block2; c) Block1, Block2 and Block3 respectively. Compare the final training errors and testing errors with LeNet in a table.

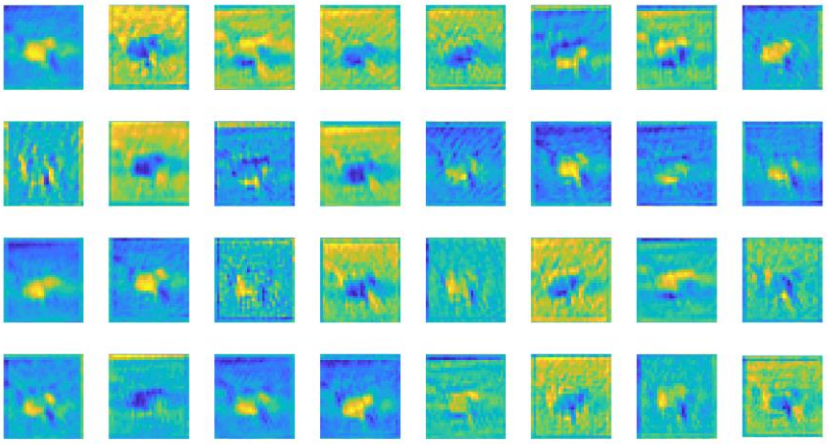
Block	(full)	(a)	(b)	(c)
Training error	0.1504	0.5027	0.5032	0.2426
Validation error	0.2287	0.5189	0.5088	0.2692
Learning rate	0.05 (15 epochs) + 0.005 (10 epochs) + 0.0005 (5 epochs)	0.005 (15 epochs) + 0.005 (10 epochs) + 0.0005 (5 epochs)	0.005 (15 epochs) + 0.005 (10 epochs) + 0.0005 (5 epochs)	0.005 (15 epochs) + 0.005 (10 epochs) + 0.0005 (5 epochs)
Block 5 filter size	1×1×64	16×16×32	8×8×32	4×4×64

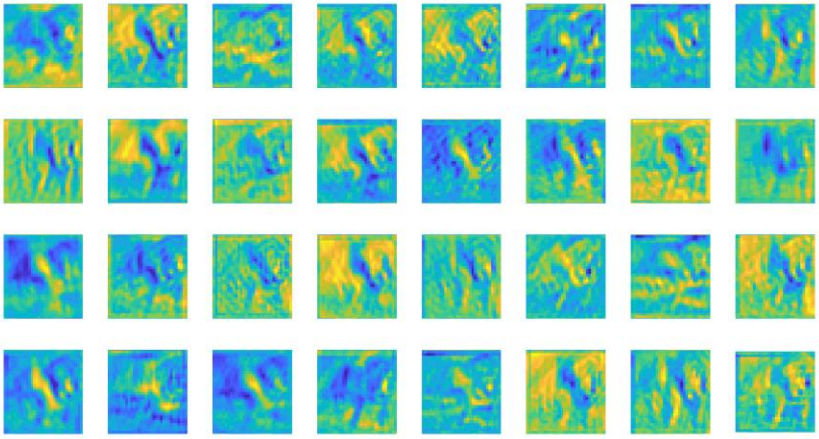
(1) I have observed that both the training and validation errors increase when I decrease the number of blocks in the network under most cases. This matches my intuition very well as a network with fewer parameters and fewer layers will naturally have a worse learning ability.
(2) But one interesting phenomenon is that both the training and testing errors of structure (a) and (b) are really close to each other, (b) is just doing a slightly better for validation accuracy. It indicates that under a same training setting, adding Block 2 to structure (a) does not improve the accuracy much.
(3) At first time when I trained structure (a), the weights failed to converge as I saw the training error remain almost a same 90%. Then I adjusted the learning rate for the first 15 epochs to be 0.1 of the original value. After this adjustment, structure (a) successfully converged. The reason is probably because the number of weights in structure (a) is much fewer than the full version network, so the original learning rate could be too big for structure (a). This makes structure (a) hard to converge during training.

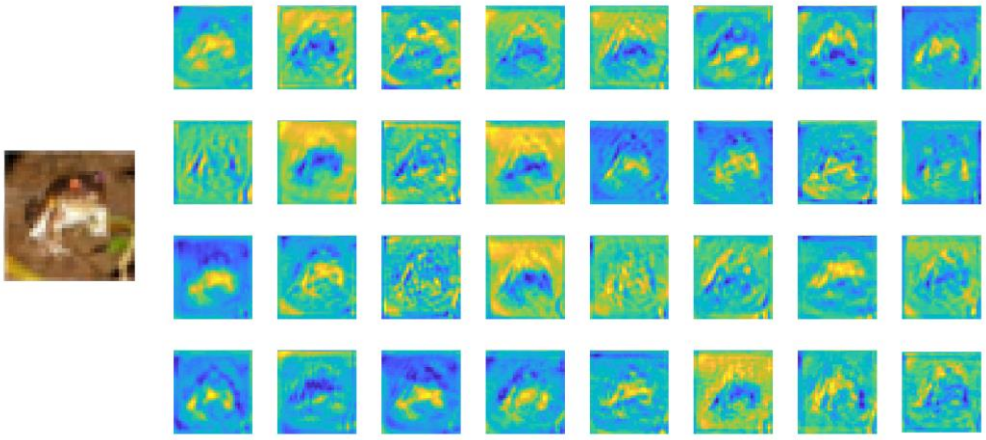
3. For each image, visualize the filter response maps of 32 filters in the first layer.

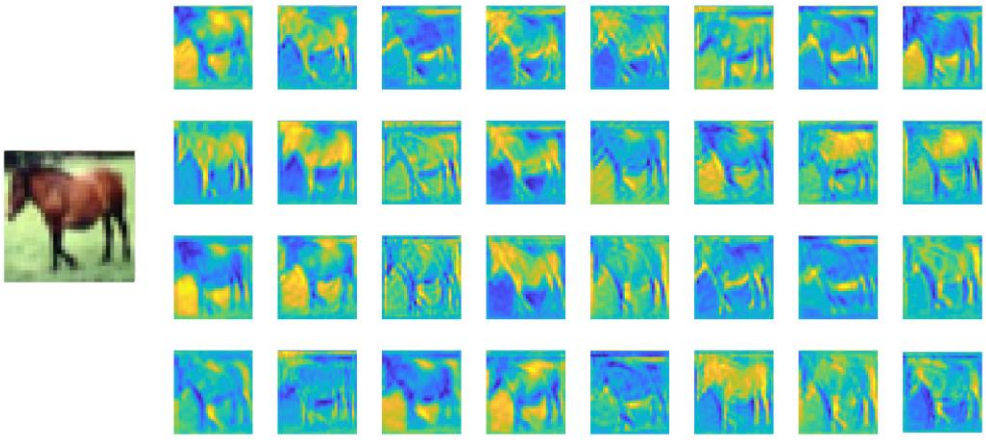


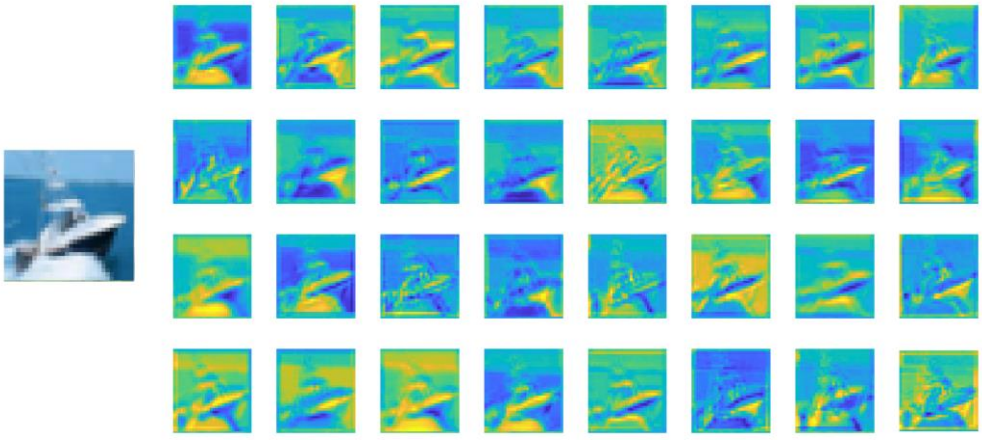


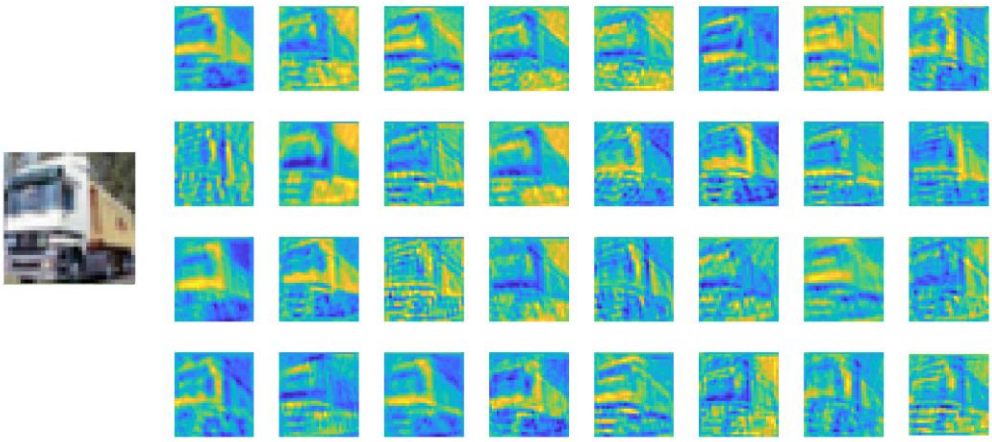












Observation:
(1) It can be clearly seen that the filter responses look quite similar to the original images. This is probably because the filters in the first layer are trying to extract some features (probably some geometric features including contour etc.) of the image so that the following layers could further extract even higher-level features based on these features.

Codes for drawing filter responses:

```
function [ ] = draw_filter_response(model, net)


index = 'images/';
for image_index = 1 : 10
    image_path = strcat('images/', num2str(image_index), '.png');
    img = imread(image_path);
    figure;
    imshow(img);
    img = single(img) - model.net.averageImage;
    res = vl_simplenn(net, img);
    responses = res(2).x;

    figure;
    for i = 1 : 32
        subplot(4,8,i);
        image(responses(:,:,i),'CDataMapping','scaled');
    end
    axis tight;
    axis off;
    set(findobj(gcf, 'type','axes'), 'Visible','off')
    daspect([1 1 1]);
end


end
```