Assignment1

Training

You can reproduce the experiment by run train.py.
The foundation of the script is PyTorch official Tutorial.
https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html
I also refer to https://pytorch.org/docs/stable/_modules/torchvision/models/resnet.html
to write my own network in

Data Augmentation:

- RandomCrop
- RandomHorizontalFlip
- Normalize

Tricks come from
https://blog.csdn.net/weixin_40123108/article/details/85246784

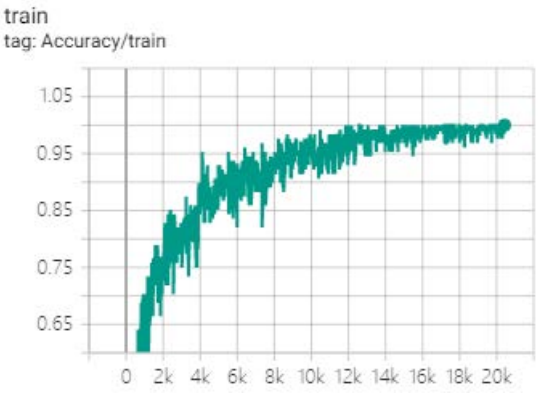
Optimizer

Adam with initial learning rate 0.001, betas=(0.9, 0.999), eps=1e-08, weight_decay=5e-4
Half the learning rate every 10 epochs
20k+ iterations in total

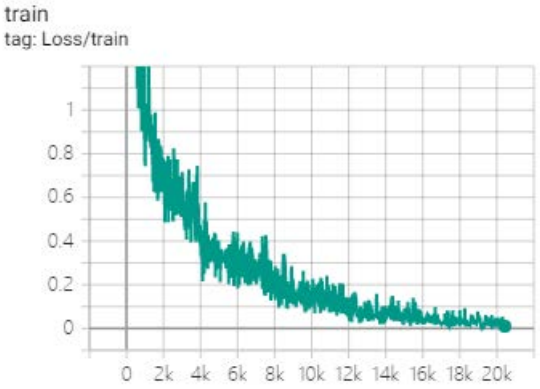
Metric

Cross_entropy

Learning Progress

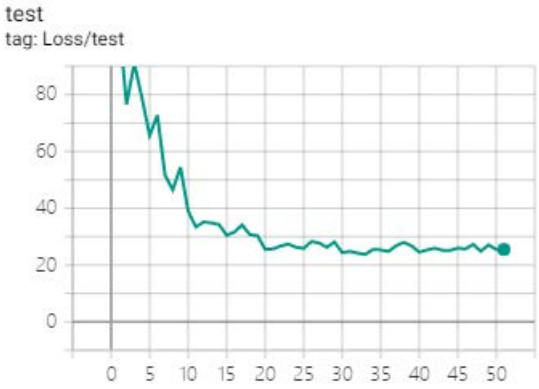


Training accuracy among mini-batch, final accuracy is 92.97% after 10k iterations



Training loss among 100 iterations, final loss is 9.8683e-3 after 20k iterations

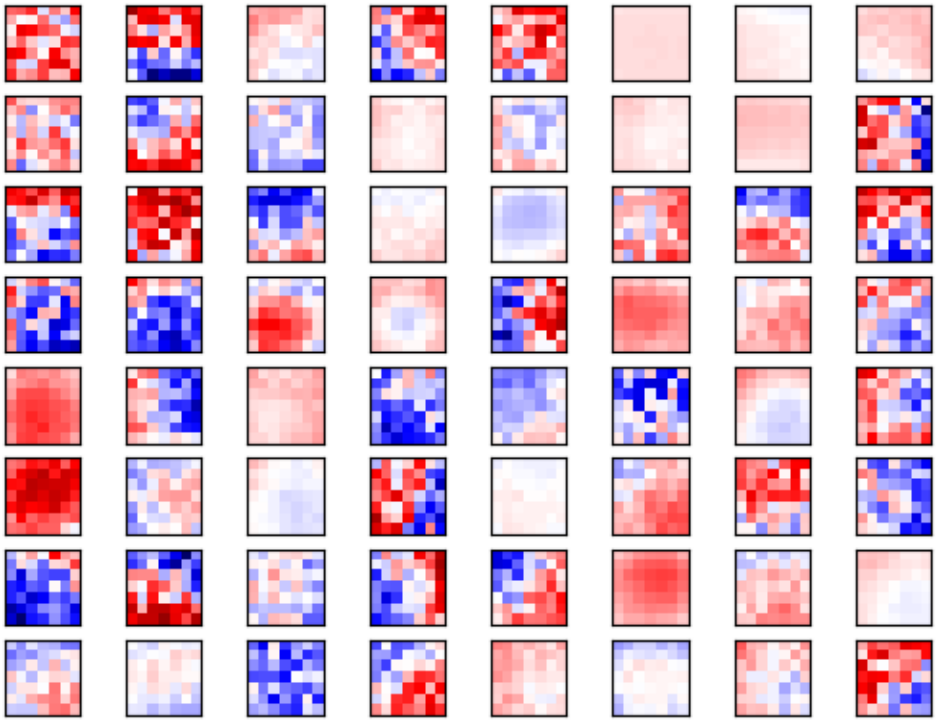
Testing accuracy among the whole test set, final accuracy is 91.2% after 20k iterations



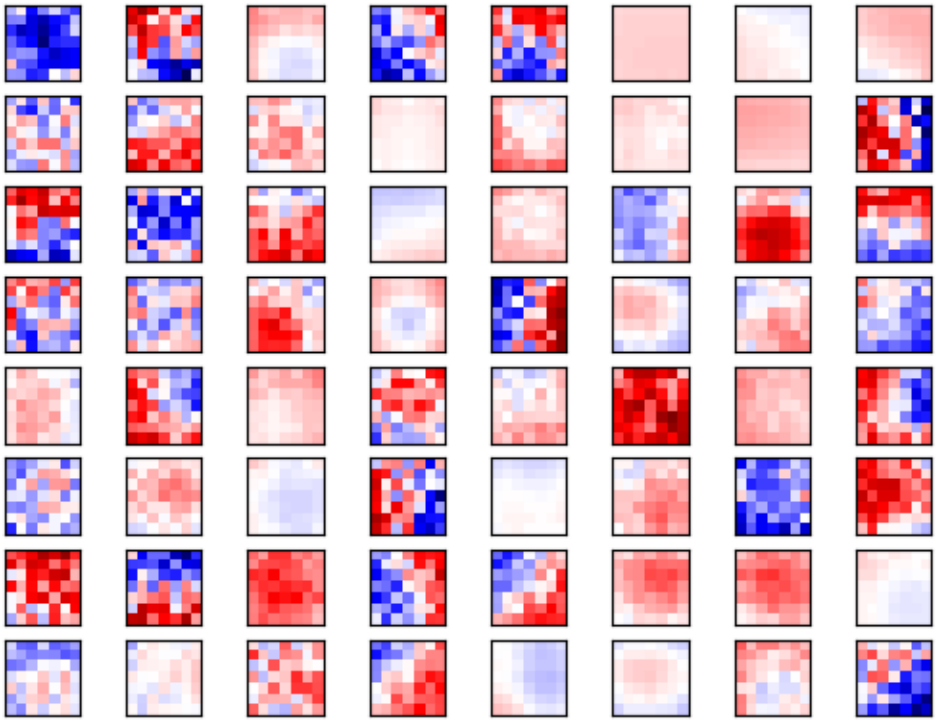
Testing loss among the whole test set, final loss is 25.49 after 20k iterations

Filter visualization

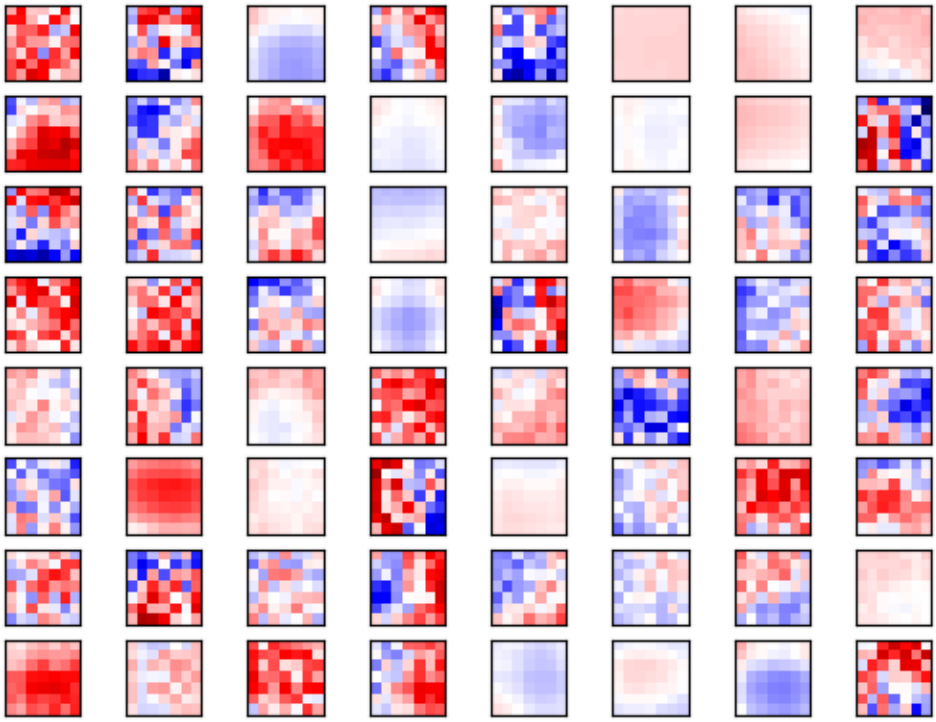
Utils.py come from https://github.com/grishasergei/conviz



Conv1-chanel 0



Conv1-chanel 1

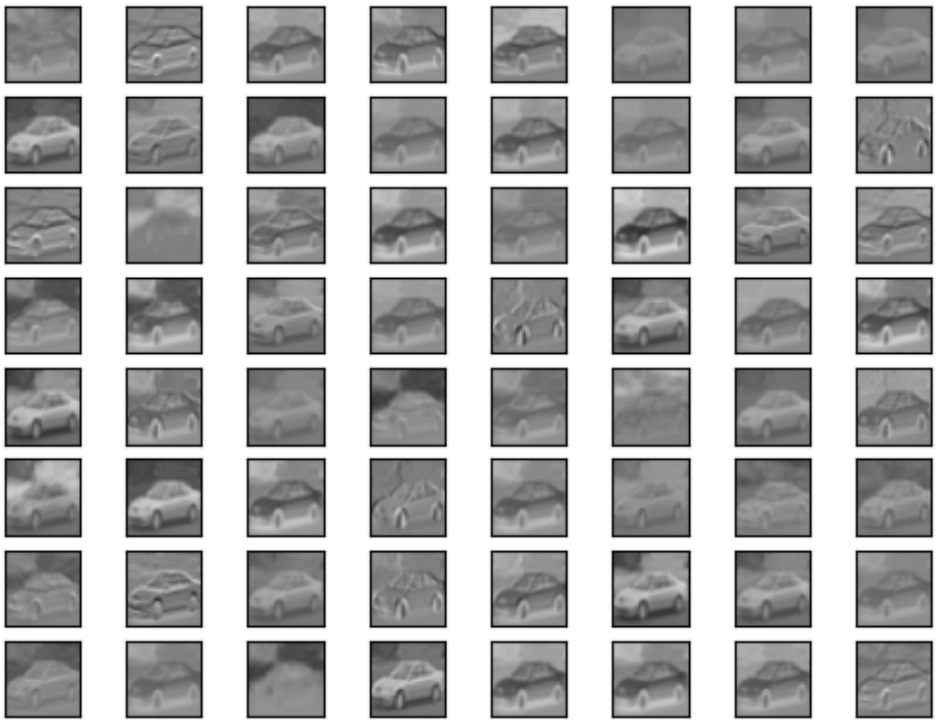


Conv1-chanel 2

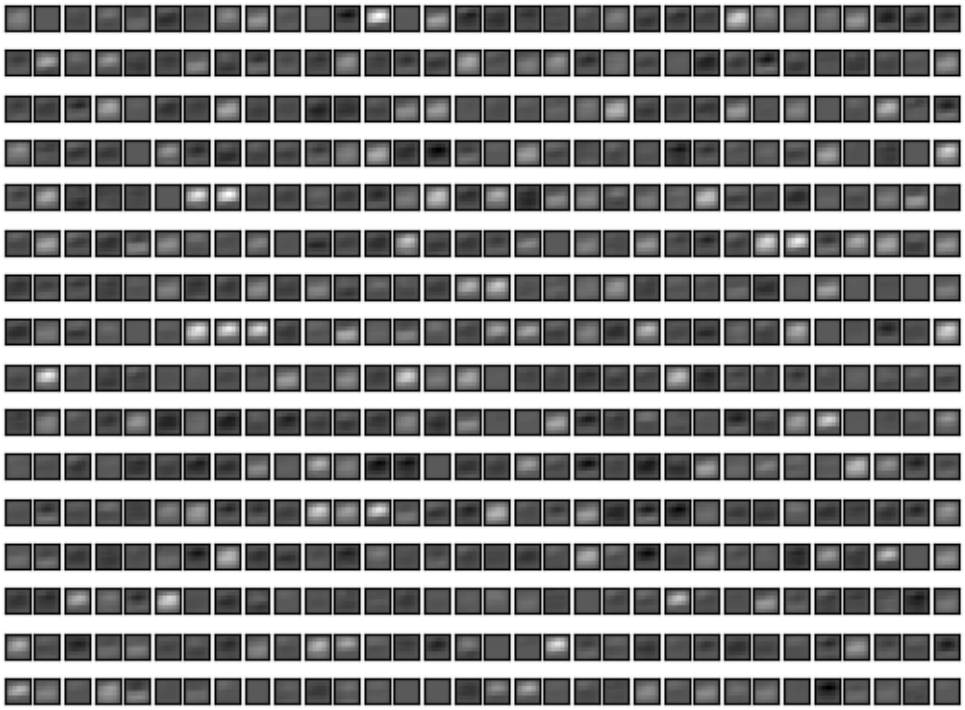
Feature mapping visualization

Visualized by a modified version of utils.py from https://github.com/grishasergei/conviz

Input image, the 7876th image in cifar10 image dataset



Conv1_output_feature_map



Conv5_last_layer_output_feature_map

Viusalization with deconvlution and max-unpool

This strategy come from https://github.com/kvfrans/feature-visualization

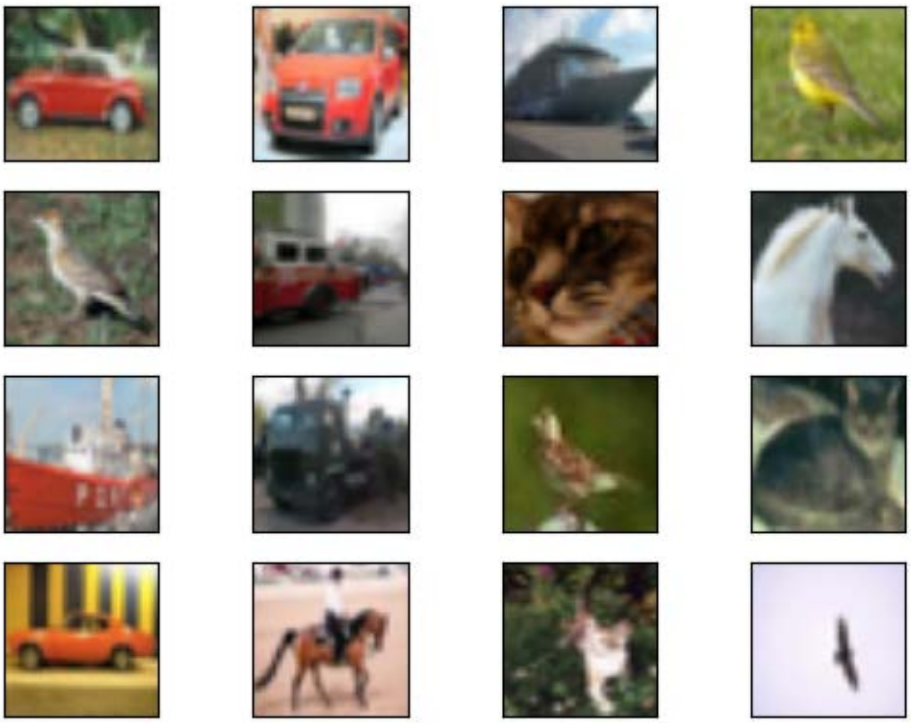
Code can be found in

- plot_conv1_feature_map.py
- plot_conv1_filters.py
- plot_conv1_reconstructed_pattern.py
- plot_conv5_feature_map.py
- plot_conv5_reconstructed_pattern_shortcut.py
- plot_conv5_reconstructed_pattern_trunk.py

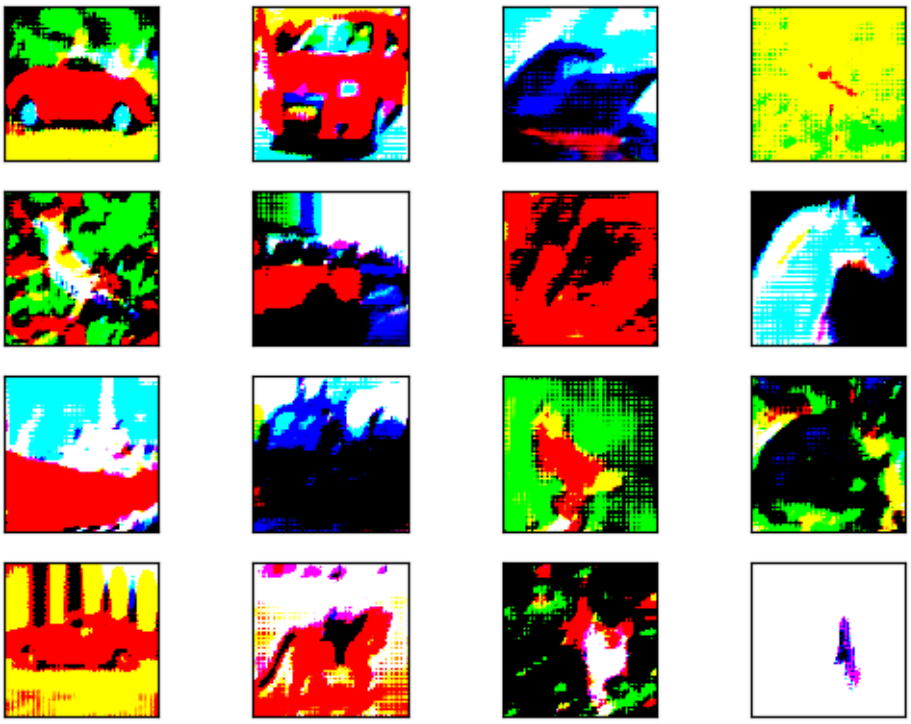
.

The process is like backward propagation. Backward propagation propagate the gradients, while we propagate the feature map by sequentially apply the inverse operator corresponding to each operator from tail to head. Note that inverse operator defined here is not strict. I Inverse operator here just recover shape, e.g. deconvolution – convolution, maxpooling-maxunpooling.

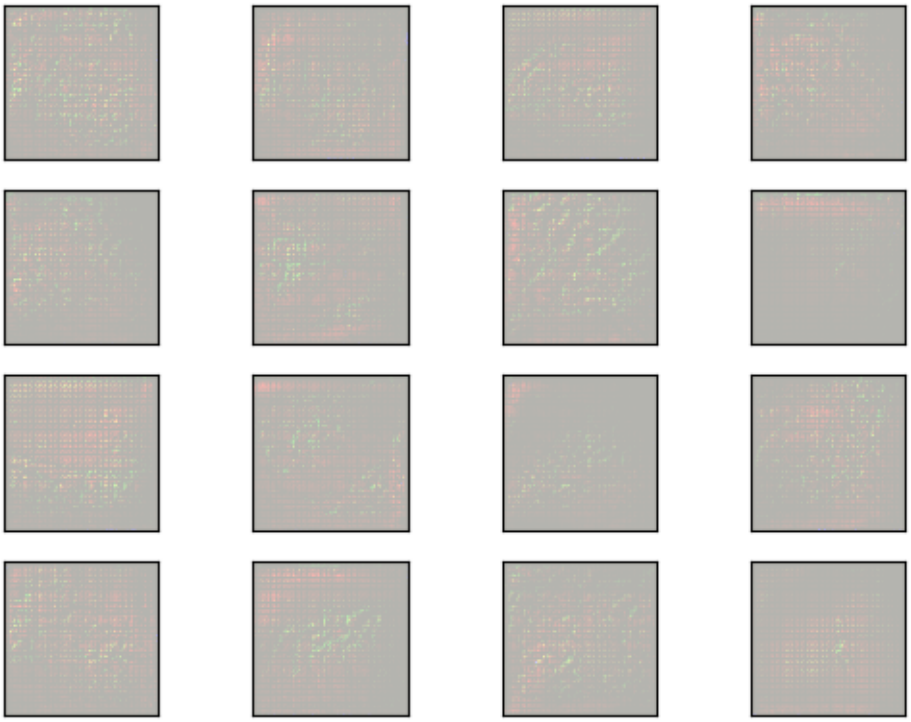
Original image(corresponding image patches)



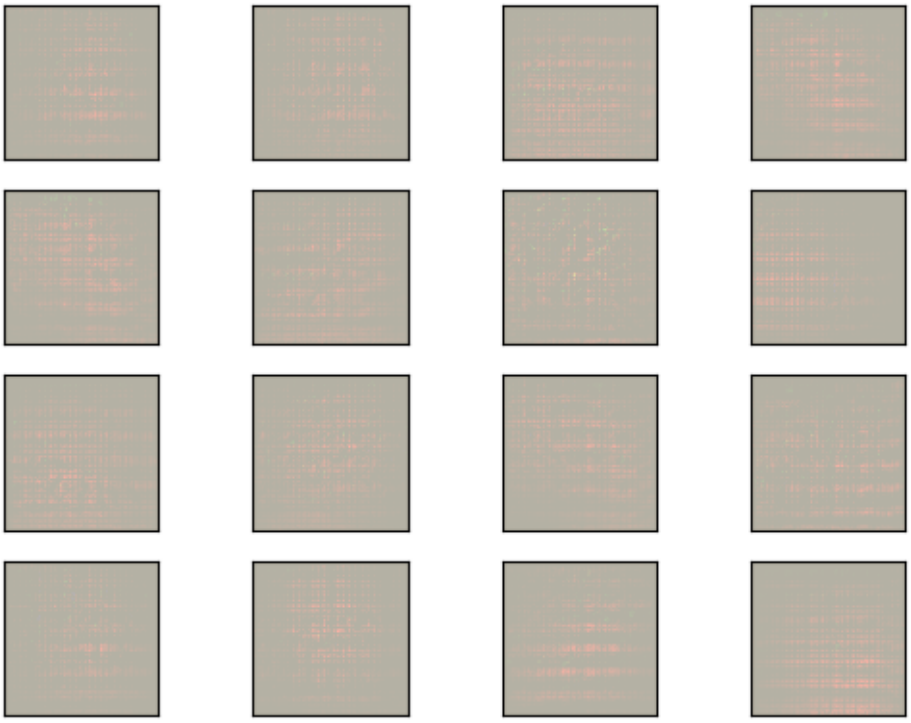
Input image, the 4531-4546th image in cifar10 image dataset



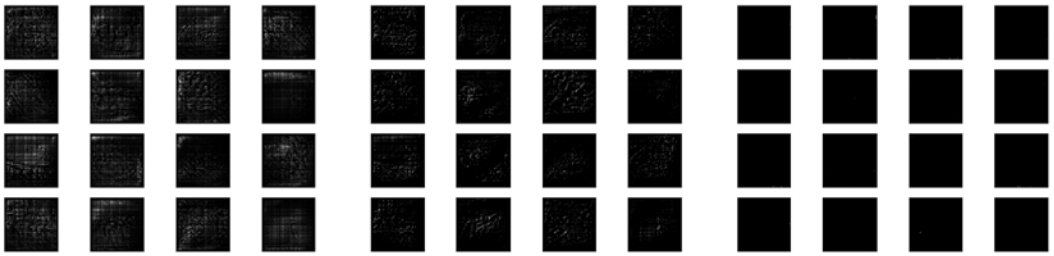
Reconstructed Pattern of Conv1



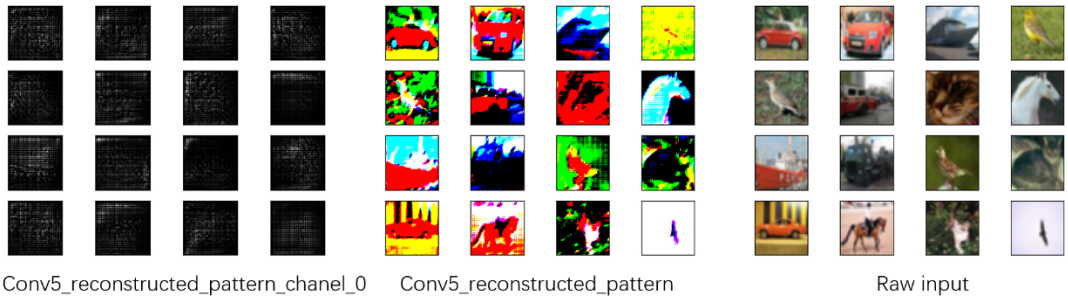
Reconstructed Pattern of Conv5 through shortcuts

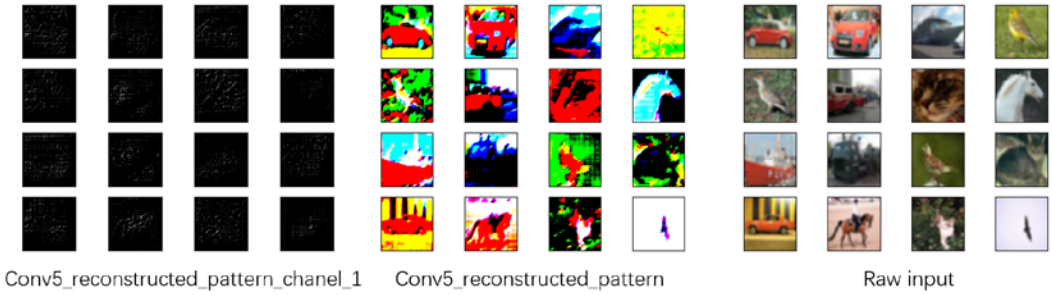


Reconstructed Pattern of Conv5 through trunk branch



Reconstructed Pattern of Conv5 through trunk branch(separated)





Viusalization with gradient based strategy

Idea and the base code come from https://github.com/utkuozbulak/pytorch-cnn-visualizations. The basic idea is to find a pattern that cause a specified filter output the maximum activation map. This can be viewed as an unconstrained optimization problem. Thus we can use gradient decent to solve it.

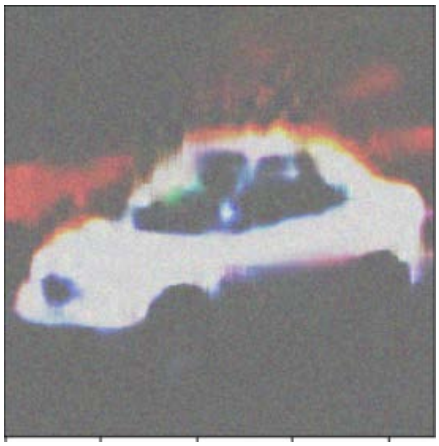
I just focus on the first filter of the last layer in the block Conv5_x. In order to get fine performance, I make two little changes. The first one to find an image in 10000 which cause specified filter output the maximum. I find the 7876th image activate the filter most.

```
#     temp_img = temp_img.to(device)
#     model(temp_img.expand(1, -1, -1, -1))
max_index = 0
max_value = 0
max_activation_image = None
max_activation_input = None
for i in trange(10000):
    temp_img, label = testloader.dataset[i]
    img_for_plot = temp_img.clone()
    temp_img = norm(temp_img)
    temp_img = temp_img.to(device)
    input = temp_img.expand(1, -1, -1, -1)
    model(input)
    print(activations.features[0, 0].sum())
    v = activations.features[0, 0].sum()
    if i == 0:
        max_value = v
    else:
        if v > max_value:
            max_index = i
            max_value = v
            max_activation_image = img_for_plot.clone()
            max_activation_input = input.clone()



print(max_index)
```

```
tensor( 4.0024, device='cuda:0', grad_fn=<SumBackward0>)
7876
```

Input image, the 7876th image in cifar10 image dataset

The second change is to maximize the activation as well as minimize the distance between the pattern we try to find and the input image. Finally, I find a nice pattern for the filter.



Reconstructed Pattern of Conv5-filter1 based on gradient

Feedback

Time: 4 + 2 + 4 + 17 hours
Comments: Prof. Chen is nice. The lecture is becoming increasingly good.
Comments: hard for everyone
Suggestion: I am looking forward to learn more basic theory and knowledge on general artificial intelligence.

Reference

https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html
https://pytorch.org/docs/stable/_modules/torchvision/models/resnet.html
https://pytorch.org/docs/stable/index.html
https://github.com/utkuozbulak/pytorch-cnn-visualizations
https://github.com/kvfrans/feature-visualization
https://github.com/grishasergei/conviz