filter-viz

September 22, 2019

0.1 Visualizing the trained filters

In [68]:
```
# some startup!
import numpy as np
import matplotlib
# This is needed to save images
matplotlib.use('Agg')
import matplotlib.pyplot as plt
import torch
```

In [101]:
```
# load the model saved by train.py
# This will be an instance of models.softmax.Softmax.
# NOTE: You may need to change this file name.
model_type = "convnet"

if model_type == "softmax":
    model = torch.load('softmax.pt')
elif model_type == "twolayernn":
    model = torch.load('twolayernn.pt')
elif model_type == "convnet":
    model = torch.load('convnet.pt')
```

In [102]:
```
# collect all the weights
w = None
##############################################################################
# TODO: Extract the weight matrix (without bias) from softmax_model, convert
# it to a numpy array with shape (10, 32, 32, 3), and assign this array to w.
# The first dimension should be for channels, then height, width, and color.
# This step depends on how you implemented models.softmax.Softmax.
##############################################################################
if model_type == "softmax":
    w = model.w.cpu()
    w = w.weight.data.numpy()
    w = np.reshape(w, (10, 32, 32, 3))
elif model_type == "twolayernn":
    w = model.w1.cpu()
    w = w.weight.data.numpy()
    w = np.reshape(w, (model.hidden_dim, 3, 32, 32))
```

```
            w = np.transpose(w, (0, 2, 3, 1))
        elif model_type == "convnet":
            w = model.conv1.cpu()
            w = w.weight.data.numpy()
            w = np.transpose(w, (0, 2, 3, 1))
        ###############################################################################
        #                               END OF YOUR CODE                              #
        ###############################################################################
        # obtain min,max to normalize
        w_min, w_max = np.min(w), np.max(w)
        # classes
        classes = ['plane', 'car', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'tr
        # init figure
        fig = plt.figure(figsize=(6,6))
        for i in range(10):
            wimg = 255.0*(w[i].squeeze() - w_min) / (w_max - w_min)
            # subplot is (2,5) as ten filters are to be visualized
            fig.add_subplot(2,5,i+1).imshow(wimg.astype('uint8'))
        # save fig!
        if model_type == "softmax":
            fig.savefig('softmax_filt.png')
        elif model_type == "twolayernn":
            fig.savefig('twolayernn_filt.png')
        elif model_type == "convnet":
            fig.savefig('convnet_filt.png')
        print('figure saved')
```

```
c:\users\kingsman142\appdata\local\programs\python\python36\lib\site-packages\matplotlib\pyplot
  max_open_warning, RuntimeWarning)
```

```
figure saved
```

In [103]:
```
        # vis_utils.py has helper code to view multiple filters in single image. Use this to
        # neural network adn convnets.
        # import vis_utils
        from vis_utils import visualize_grid
        # saving the weights is now as simple as:
        if model_type == "softmax":
            plt.imsave('softmax_gridfilt.png',visualize_grid(w, padding=3).astype('uint8'))
        elif model_type == "twolayernn":
            plt.imsave('twolayernn_gridfilt.png',visualize_grid(w, padding=3).astype('uint8'
        elif model_type == "convnet":
            plt.imsave('convnet_gridfilt.png',visualize_grid(w, padding=3).astype('uint8'))
        # padding is the space between images. Make sure that w is of shape: (N,H,W,C)
        print('figure saved as a grid!')
```

```
figure saved as a grid!
```

In [ ]: