In [1]:

```
import numpy as np
import torch
from torch.autograd import Variable
import torchvision
import PIL

from style_modules import ContentLoss, StyleLoss, TotalVariationLoss
from style_utils import features_from_img, extract_features, rel_error, style_transfer
from image_utils import preprocess
```

In [2]:

```
def content_loss_test(correct, content_loss):
    content_image = 'styles_images/tubingen.jpg'
    image_size = 192
    content_layer = 3
    content_weight = 6e-2

    c_feats, content_img_var = features_from_img(content_image, image_size, cnn, dtype)

    bad_img = Variable(torch.zeros(*content_img_var.data.size()))
    feats = extract_features(bad_img, cnn)

    student_output = content_loss(content_weight, c_feats[content_layer], feats[content_layer]).data.numpy()
    error = rel_error(correct, student_output)
    print('Content Loss Maximum error is {:.3f}'.format(error))

def gram_matrix_test(correct, style_loss):
    style_image = 'styles_images/starry_night.jpg'
    style_size = 192
    feats, _ = features_from_img(style_image, style_size, cnn, dtype)
    student_output = style_loss.gram_matrix(feats[5].clone()).data.numpy()
    error = rel_error(correct, student_output)
    print('Gram Matrix Maximum error is {:.3f}'.format(error))


def style_loss_test(correct, style_loss):
    content_image = 'styles_images/tubingen.jpg'
    style_image = 'styles_images/starry_night.jpg'
    image_size = 192
    style_size = 192
    style_layers = [1, 4, 6, 7]
    style_weights = [300000, 1000, 15, 3]
```

```
    c_feats, _ = features_from_img(content_image, image_size, cnn, dtype)
    feats, _ = features_from_img(style_image, style_size, cnn, dtype)
    style_targets = []
    for idx in style_layers:
        style_targets.append(style_loss.gram_matrix(feats[idx].clone()))

    student_output = style_loss(c_feats, style_layers, style_targets, style_weights).data.numpy()
    error = rel_error(correct, student_output)
    print('Style Loss Error is {:.3f}'.format(error))


def tv_loss_test(correct, tv_loss):
    content_image = 'styles_images/tubingen.jpg'
    image_size = 192
    tv_weight = 2e-2

    content_img = preprocess(PIL.Image.open(content_image), size=image_size)
    content_img_var = Variable(content_img.type(dtype))

    student_output = tv_loss(content_img_var, tv_weight).data.numpy()
    error = rel_error(correct, student_output)
    print('TV Loss Error is {:.3f}'.format(error))
```

In [3]:

```
dtype = torch.FloatTensor
# Uncomment out the following line if you're on a machine with a GPU set up for PyTorch!
# dtype = torch.cuda.FloatTensor

cnn = torchvision.models.squeezenet1_1(pretrained=True).features
cnn.type(dtype)

# Fix the weights of the pretrained network
for param in cnn.parameters():
    param.requires_grad = False

answers = np.load('data/style-transfer-checks.npz') # for local test
```

Test content loss (1pt)

In [4]:

```
content_loss = ContentLoss()
content_loss_test(answers['cl_out'], content_loss) # should print out 0.0
```

```
Content Loss Maximum error is 0.000
```

Test style loss (2pt)

In [5]:
```
style_loss = StyleLoss()
gram_matrix_test(answers['gm_out'], style_loss) # should print out 0.0
style_loss_test(answers['sl_out'], style_loss) # should print out 0.0
```

```
Gram Matrix Maximum error is 0.000
Style Loss Error is 0.000
```

Test total variation loss (1pt)

In [6]:
```
tv_loss = TotalVariationLoss()
tv_loss_test(answers['tv_out'], tv_loss) # should print out 0.0
```

```
TV Loss Error is 0.000
```