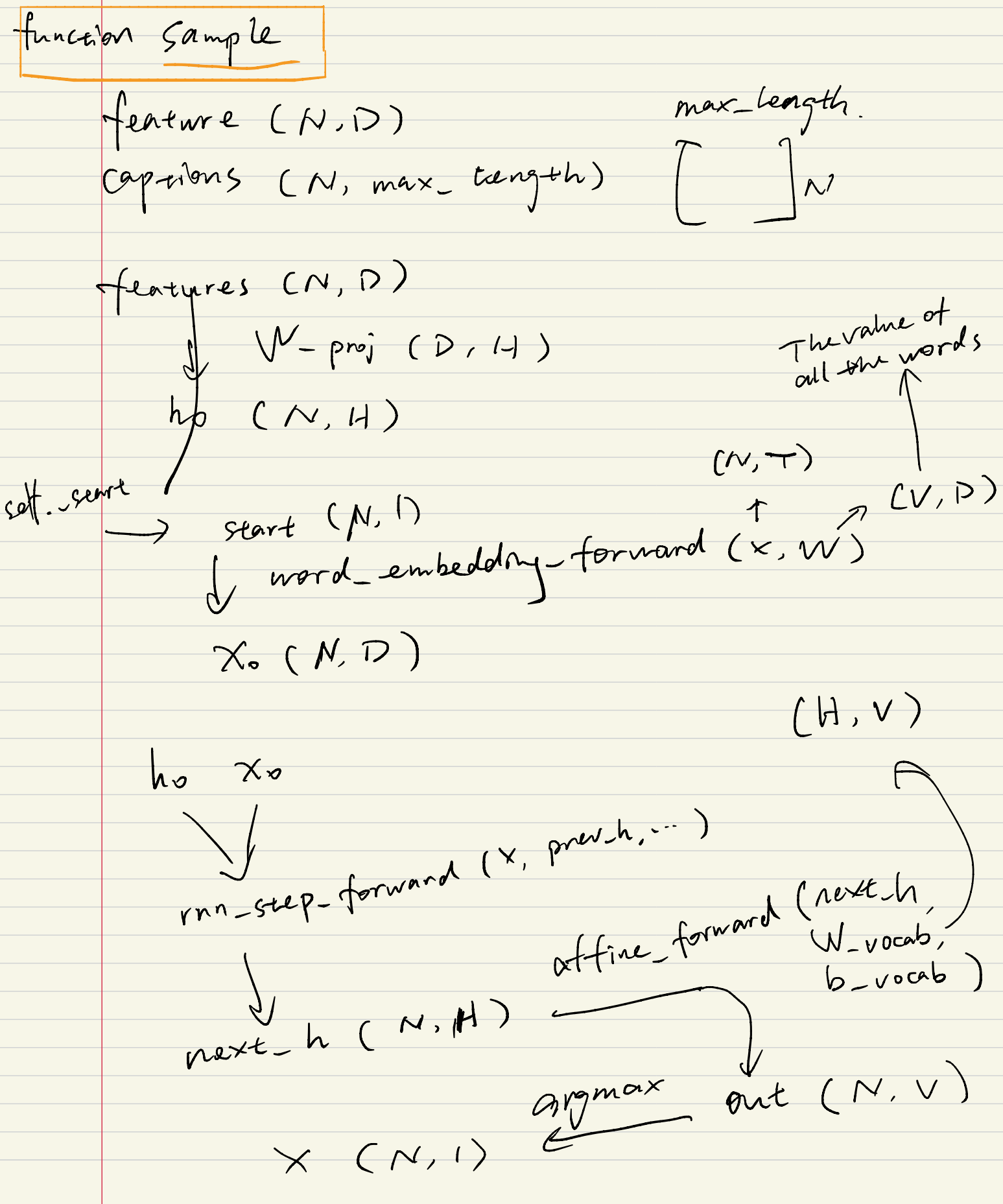
CS231n Assignment 3

- RNN & LSTM Image captioning
- Network Visualization
  - Saliency maps
  - Class Visualization
  - Fooling Images
- Style Transfer

  Aside: Matrix Norm and Vector Norm
- Generative Adversarial Networks

CS231n Assignment 3

function sample



LSTM_Captioning

$a = W_x x_t + W_h h_{t-1} + b$

(N, D) $(D, 4H)$ (N, H) $(H, 4H)$

$\downarrow$

$(N, 4H)$

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

a

$D \left[\; | \; | \; | \; | \; \right]$ $4H$

$i \quad f \quad o \quad g$

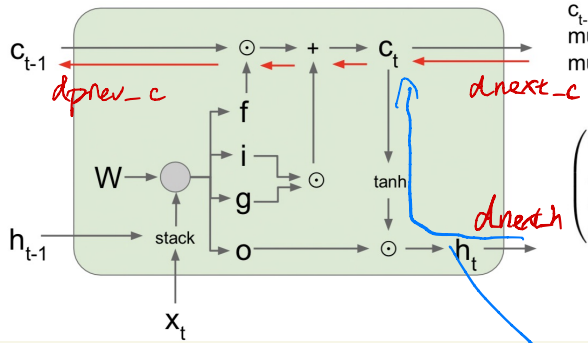
$c_t = f \odot c_{t-1} + i \odot g$

$h_t = o \odot \tanh(c_t)$

Forward pass

Long Short Term Memory (LSTM): Gradient Flow

[Hochreiter et al., 1997]



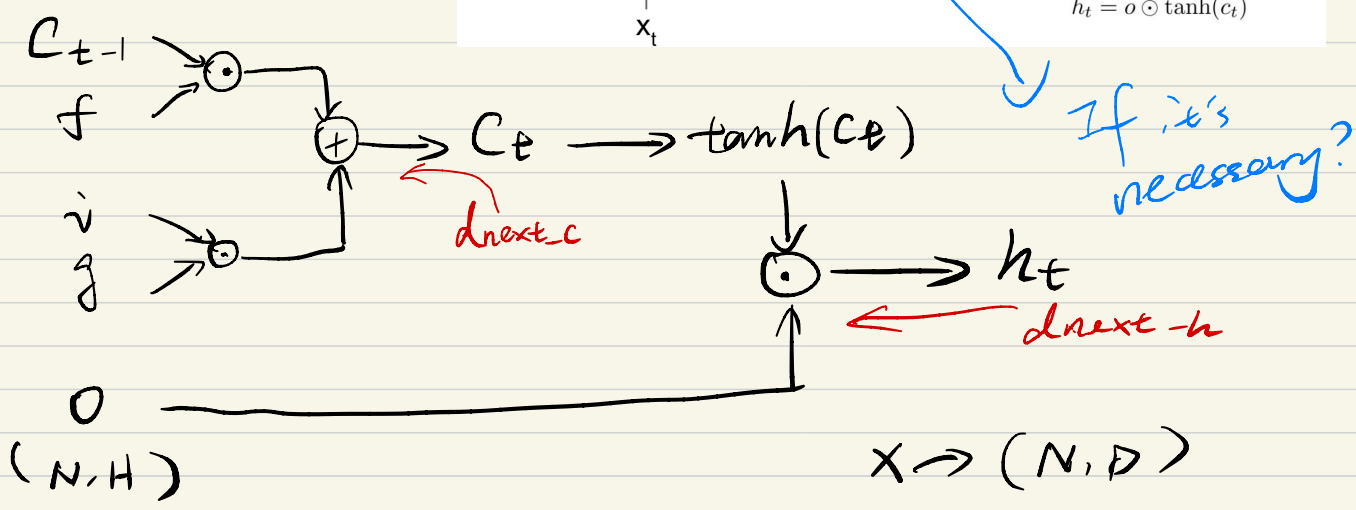
Backpropagation from c_t to c_{t-1} only elementwise multiplication by f, no matrix multiply by W

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

Backward pass



input → dnext_h , dnext_c

$[N,D] \cdot [D,4H]$

$[x] \cdot [W_x]$

$Y = xW$

$dY = (x)\,dW$

$\frac{\partial Y}{\partial W} = (x)^T$

$dY = dx \cdot W$

$tr(dY) = tr(W^T dx^T)$

$\frac{\partial Y}{\partial x^T} = W \Rightarrow \frac{\partial Y}{\partial x} = W^T$

$$do = dnext\_h \odot \tanh(c_t)$$

$$dW_{xo} = x^T [do \odot (o \odot (1-o))]$$

(D, N) (N, H) ③

$$df = dnext\_c\_all \odot prev\_c$$

$$dW_{xf} = x^T [df \odot f \odot (1-f)]$$

②

$$di = dnext\_c\_all \odot g$$

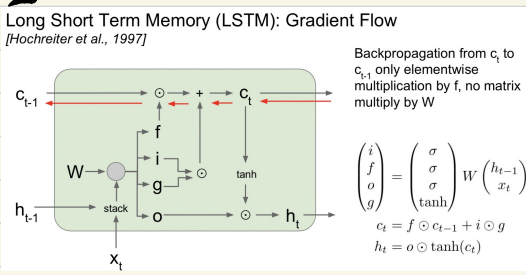
$$dW_{xi} = x^T [di \odot i \odot (1-i)]$$

①

$$dg = dnext\_c\_all \odot i \quad \tanh'(x W_{xg})$$

$$dW_{xg} = x^T [dg \odot (1-g^2)]$$

④



这里需不需要考虑 dnext_h? YES

$\tanh'(next\_c)$

$$dnext\_c\_all = dnext\_c + dnext\_h \odot o \odot (1 - \tanh^2(next\_c))$$

$$dprev\_c = dnext\_c\_all \odot f$$

①②③④ 前面都乘 x^T 再得到 dWx，可以 concatenate 起来一起乘

$$dW_x = x^T \cdot [①\ ②\ ③\ ④]$$

(D, N) dgate (N, 4H)

$$dW_h = prev\_h^T \cdot dgate$$

$$dx = dgate \cdot W_x^T$$

$$dprev\_h = dgate \cdot W_h^T$$

(N, 4H)

$$db = np.sum(dgate, axis=0)$$

Network Visualization

1. **Saliency Maps:** Saliency maps are a quick way to tell which part of the image influenced the classification decision made by the network.
2. **Fooling Images:** We can perturb an input image so that it appears the same to humans, but will be misclassified by the pretrained network.
3. **Class Visualization:** We can synthesize an image to maximize the classification score of a particular class; this can give us some sense of what the network is looking for when it classifies images of that class.

create a Session that dynamically allocates memory.
↳ allowing gpu memory growth.

默认情况下，TensorFlow 会映射进程可见的所有 GPU 的几乎所有 GPU 内存（取决于 CUDA_VISIBLE_DEVICES）。通过减少内存碎片，可以更有效地使用设备上相对宝贵的 GPU 内存资源。

memory fragmentation.
a phenomenon in which storage space is used inefficiently, reducing capacity or performance and often both.

```
config = tf.ConfigProto()
config.gpu_options.allow_growth = True
session = tf.Session(config=config, ...)
```

tf.get_variable(name, shape=[..], ...)
创建一个 variable. 默认的 dtype 为 tf.float32, 默认的初始化通过 tf.glorot_uniform_initializer 随机得到.

tf.variable_scope(name, reuse=..) 可以将 tf.Variable 对象隐式地封装. 即在不同的 scope 下可选择重新使用现有变量还是创建新变量.

```
def my_image_filter(input_images):
    with tf.variable_scope("conv1"):
        # Variables created here will be named "conv1/weights", "conv1/biases".
        relu1 = conv_relu(input_images, [5, 5, 32, 32], [32])
    with tf.variable_scope("conv2"):
        # Variables created here will be named "conv2/weights", "conv2/biases".
        return conv_relu(relu1, [5, 5, 32, 32], [32])
```

tf.Variable

这里 "conv_relu" 为创建一个卷积层的函数. 其中包括创建 W和b. 由于在不同的层时都需要创建新的变量, 于是用不同的 scope 隐式地进行封装.
如果要共享变量, 则可将 "reuse=True" 并保持 scope 名相同. 或调用 scope.reuse_variables()

```
with tf.variable_scope("model"):
  output1 = my_image_filter(input1)
with tf.variable_scope("model", reuse=True):
  output2 = my_image_filter(input2)
```

```
with tf.variable_scope("model") as scope:
  output1 = my_image_filter(input1)
  scope.reuse_variables()
  output2 = my_image_filter(input2)
```

Saliency Maps

$\frac{\partial \text{score}}{\partial \text{pixels of img}}$

Compute the gradient of the unnormalized score corresponding to the correct class w.r.t the pixels of the image.

image (H, W, 3) gradient (H, W, 3) → saliency map (H, W) (max axis = 2)

Class Visualization

By starting with a random noise image and performing gradient ascent on a target class, we can generate an image that the network will recognize as the target class.

$$I^* = \arg\max_I \left(S_y(I) - R(I) \right)$$

regularizer, $R(I) = \lambda \|I\|_2^2$

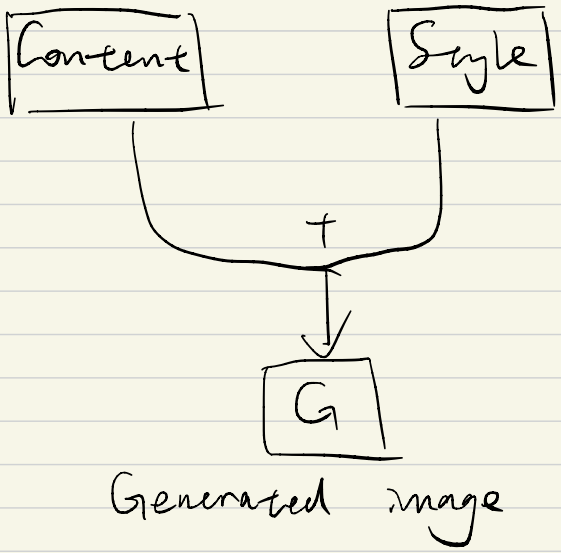
I^*: generated image

$S_y(I)$: score that a CNN assigns to image I for class y (raw unnormalized scores)

Fooling Image

Given an image and a target class, perform gradient ascent over the image to maximize the target class, stopping when the network classifies the image as the target class.

Style Transfer



Cost function:

$$\mathcal{L}(G) = \alpha \mathcal{L}_{content}(C, G) + \beta \mathcal{L}_{style}(S, G)$$

Find the generated image G:

(1) Initiate G randomly

(2) Use gradient descent to minimize $\mathcal{L}(G)$

update the pixels in G

$$G := G - \frac{\partial}{\partial G}\mathcal{L}(G)$$

Content Loss

Content loss measures how much of the generated image differs from the feature maps of the source image.

选取 hidden layer L, feature maps $A^L \in \mathbb{R}^{1 \times H_L \times W_L \times C_L}$

reshape → feature map for current image $F^L \in \mathbb{R}^{M_L \times C_L}$

feature maps for content source $P^L \in \mathbb{R}^{M_L \times C_L}$

$M_L = H_L \times W_L$

N_L $\begin{bmatrix} | & | & | & | \\ | & | & | & | \end{bmatrix}$ C_L

在计算 Content loss 时不是特别必要将 feature map reshape 成一维，为了后面计算 style loss 方便。

Each column of F^L or P^L represents the vectorized activations of a particular filter, convolved over all positions of the image.

$$\mathcal{L}_{content}(C, G) = \|F^L - P^L\|_2^2 = \sum_{ij} (F_{ij}^L - P_{ij}^L)^2$$

Style loss

Define style as correlation between activations across channels.

Aside:

Matrix Norm

① Matrix norms induced by vector norms.

(induced norm)

$A \to \mathbb{R}^{m \times n}$

$x \to \mathbb{R}^n$

$$\|A\|_p = \sup_{x \neq 0} \frac{\|Ax\|_p}{\|x\|_p}$$

这里是常规的向量的 L_p 范数

Special cases:

$$\|A\|_1 = \max_{1 \le j \le n} \sum_{i=1}^{m} |a_{ij}|$$ → maximum column sum (sum across rows)

$$\|A\|_\infty = \max_{1 \le i \le m} \sum_{j=1}^{n} |a_{ij}|$$ → maximum row sum (sum across columns)

$$\|A\|_2 = \sigma_{max}(A) = \sqrt{\lambda_{max}(A^* A)}$$ → largest singular value of matrix A.

spectral norm (the operator norm induced by the vector 2-norm)

$\|x\|_1 = \sum_i |x_i|$

$\|x\|_\infty = \max_i |x_i|$

② "Entrywise" matrix norms

treat an $m \times n$ matrix as a vector of size $m \cdot n$, and use vector norms:

$$\|A\|_p = \|\mathrm{vec}(A)\|_p = \left(\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^p \right)^{\frac{1}{p}}$$

$$\|A\|_{p,q} = \left[\sum_{j=1}^{n} \left(\sum_{i=1}^{m} |a_{ij}|^p \right)^{\frac{q}{p}} \right]^{\frac{1}{q}}$$

Special cases:

- Frobenius norm $\longrightarrow$ $\|A\|_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2} = \sqrt{\mathrm{trace}(A^* A)}$

$$= \sqrt{\sum_{i=1}^{\min\{m,n\}} \sigma_i^2(A)}$$

$\sigma_i(A)$ are the singular values of A.

induced norm: $\|A\|_2 = \sigma_{\max}(A) \leq \|A\|_F$ (Equality holds only if A is a rank-one matrix or a zero matrix)

雖然 Frobenius 範數是 $p=q=2$ 的情況.
但因爲不是 induced norm 區分, 寫成 $\|\cdot\|_F$

- Max norm $\longrightarrow$ $\|A\|_{max} = \max_{i,j} |a_{ij}|$

③ Schatten norms

The schatten p-norms arise when applying the p-norm to the vector of singular values of a matrix.

$$\|A\|_p = \left(\sum_{i=1}^{\min\{m,n\}} \sigma_i^p(A) \right)^{\frac{1}{p}}$$

Special cases:

- $p = 1$ $\longrightarrow$ nuclear norm (trace norm)
- $p = 2$ $\longrightarrow$ Frobenius norm
- $p = \infty$ $\longrightarrow$ Spectral norm

Vector Norm

$$\|x\|_p = \left(\sum_i |x_i|^p\right)^{\frac{1}{p}}$$

$\|x\|_1 = \sum_i |x_i|$ → L^1范数，当求总和问题中零和非零元素之间差异非常重要时会使用L^1范数

$\|x\|_\infty = \max_i |x_i|$

$\|x\|_2 = \sqrt{\sum_i |x_i|^2}$ → L^2范数（Euclidean Norm）

$\|x\|_2^2 = \sum_i |x_i|^2 = x^T x$

L^2范数对于元素也经常用来衡量向量的大小，但在原点附近增长缓慢，因为平方L^2范数对于中每个元素的导数只取决于对应元素本身。

前面提到了style定义为"correlations between activations across channels"因为不同的channels对应着不同的filters.

这里在实际应用中使用了Gram matrix作为covariance matrix的近似。我们希望的是generated image的activations不同channels之间的相关性和style image的相近。

Covariance matrix → $Cov[X, X]$

$$= E[(x-\mu_x)(x-\mu_x)^T]$$

$$= E[xx^T] - \mu_x \mu_x^T$$

Gram matrix → xx^T （这里的matrix X定义为 $\begin{bmatrix} — x_1^T — \\ — x_2^T — \\ \vdots \\ — x_n^T — \end{bmatrix}$）

对于向量来说Gram matrix就是内积

由于assignment中定义的F为 $\left[\underset{C_L}{|||}\right] N_L$，所以这里：

$$G_g^L = F^T F \qquad G_s^L = S^T S$$

$$\mathcal{L}_{style}(S, G) = \|G_g^L - G_s^L\|_2^2 = \sum_{ij}(G_{g_{ij}}^L - G_{s_{ij}}^L)^2$$

使用 channels 间协相关矩阵作为 style 的衡量标准的解释为，可表示哪些高层的纹理或元素倾向于同时或不同时出现的程度.

Total-variation Regularization.

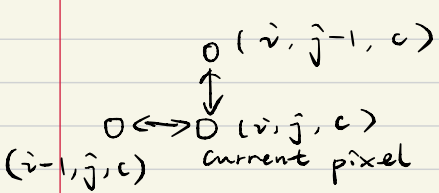
It's also helpful to encourage "smoothness" in the images.

做法是在 loss function 中加入对于 wiggles 或 "total variation" 的 penalty

total-variation

↓

sum of the squares of differences in the pixel values for all pairs of pixels that are next to each other (horizontally or vertically).



img: (1, H, W, 3)

$$\Rightarrow \text{tf.reduce\_sum}\left((\text{img}[:,1:,:,:]-\text{img}[:,:-1,:,:])^2\right)+\text{tf.reduce\_sum}\left((\text{img}[:,:,1:,:]-\text{img}[:,:,:-1,:])^2\right)$$

Generative Adversarial Networks (GANs)

Aside: 交叉熵和最大似然估计.

① 最大似然估计

一组含有 m 个样本的数据集 $X=\{x^{(1)},\dots,x^{(m)}\}$，独立地由未知的真实数据分布 $P_{data}(x)$ 生成.

令 $P_{model}(x;\theta)$ 是由 θ 确定在相同空间上的概率分布. $P_{model}(x;\theta)$ 将任意输入 x 映射到实数来估计真实概率 $P_{data}(x)$.

对 θ 的最大似然估计定义为 $\rightarrow$

$$\theta_{ML}=\arg\max_{\theta} P_{model}(X;\theta)$$
$$=\arg\max_{\theta}\prod_{i=1}^{m} P_{model}(x^{(i)};\theta)$$

最大似然估计是找到使得模型可产生出实际样本数据的可能性最大化的模型参数的方法.

为了简化计算，同时保证数值稳定，将求积转化为计算求和的形式.

$$\theta_{ML} = \arg\max_{\theta} \sum_{i=1}^{m} \log P_{model}(x^{(i)}; \theta)$$

因为当重新缩放代价函数 argmax 不变，可以除以 m 得到以训练数据经验分布 $\hat{P}_{data}$ 相关的期望作为准则：

$$\theta_{ML} = \arg\max_{\theta} E_{x \sim \hat{P}_{data}} \log P_{model}(x; \theta)$$

Population → Observation → model

$P_{data}(x)$ $\quad$ $\hat{P}_{data}(x)$ $\quad$ $P_{model}(x; \theta)$

从 population 到 observation 再到 model 的关系流程如下：

最大似然估计可以看成是一种试图使最小化训练集上经验分布 $\hat{P}_{data}$ 和模型分布之间的差异。

这种差异可通过 KL 散度度量：

$$D_{KL}(\hat{P}_{data} \| P_{model}) = E_{x \sim \hat{P}_{data}} [\log \hat{P}_{data}(x) - \log P_{model}(x)]$$

$$\min D_{KL}(\hat{P}_{data} \| P_{model}) = \min - E_{x \sim \hat{P}_{data}} [\log P_{model}(x)]$$

最小化 KL 散度其实就是在最小化分布之间的交叉熵。任何一个由负对数似然组成的损失都是定义在训练集经验分布和定义在模型上的概率分布之间的交叉熵。

(2) 交叉熵

自信息 $I(x) = -\log P(x)$ → 事件为 x 的信息量

香农熵：$H(x) = E_{x \sim P}[I(x)] = -E_{x \sim P}[\log P(x)]$

→ 遵循分布 P 的事件产生的期望信息量

The Shannon entropy of a distribution is the expected amount of information in an event drawn from that distribution.

KL散度： $D_{KL}(P\|Q) = \underset{x\sim p}{E}\left[\log \frac{P(x)}{Q(x)}\right] = \underset{x\sim p}{E}\left[\log P(x) - \log Q(x)\right]$

↘ 衡量随机变量 x 同个概率分布 $p(x)$ 和 $Q(x)$ 的差异.

交叉熵： $H(p,Q) = H(p) + D_{KL}(P\|Q)$

$= -\underset{x\sim p}{E}\left[\log Q(x)\right] = -\sum_x p(x)\log Q(x)$

针对 Q 最小化交叉熵等价于最小化 KL 散度.