

10.13 工作日记

1. 学习了python正则表达式机制。
2. 继续研读Gatys的工作的源码,以及稍微修改参数后观看运行结果。研究了每个argparse类里的参数对于全部文件的作用
3. 方向论文 (AttnGAN Fine-grained text to image generation with attentional generative adversarial networks)

1 正则语法:

语法	说明	表达式实例	完整匹配的字符串	
字符				
一般字符	匹配自身	abc	abc	
.	匹配任意除换行符"\n"外的字符。 在DOTALL模式中也能匹配换行符。	a.c	abc	
\	转义字符,使后一个字符改变原来的意思。 如果字符串中有字符*需要匹配,可以使用/*或者字符集[*]。	a\c a\\c	a.c a\c	
[...]	字符集(字符类)。对应的位置可以是字符集中任意字符。 字符集中的字符可以逐个列出,也可以给出范围,如[abc]或[a-c]。第一个字符如果是^则表示取反,如[^abc]表示不是abc的其他字符。 所有的特殊字符在字符集中都失去其原有的特殊含义。在字符串集中如果要使用]、-或^,可以在前面加上反斜杠,或把]-放在第一个字符,把^放在非第一个字符。	a[bcde]e	abe ace ade	
预定义字符集(可以写在字符集[...]中)				
\d	数字:[0-9]	a\dc	a1c	
\D	非数字:[^\d]	a\Dc	abc	
\s	空白字符:[<空格>\t\r\n\f\v]	a\s c	a c	
\S	非空白字符:[^\s]	a\S c	abc	
\w	单词字符:[A-Za-z0-9_]	a\wc	abc	
\W	非单词字符:[^\w]	a\Wc	a c	
数量词(用在字符或(...)之后)				
*	匹配前一个字符0或无限次。	abc*	ab abccc	
+	匹配前一个字符1次或无限次。	abc+	abc abccc	
?	匹配前一个字符0次或1次。	abc?	ab abc	
{m}	匹配前一个字符m次。	ab{2}c	abbc	
{m,n}	匹配前一个字符m至n次。 m和n可以省略:若省略m,则匹配0至n次;若省略n,则匹配m至无限次。	ab{1,2}c	abc abbc	
*? +? ?? {m,n}?	使 * + ? {m,n} 变成非贪婪模式。	示例将在下文介绍。		
边界匹配(不消耗待匹配字符串中的字符)				
^	匹配字符串开头。 在多行模式中匹配每一行的开头。	^abc	abc	
\$	匹配字符串末尾。 在多行模式中匹配每一行的末尾。	abc\$	abc	
\A	仅匹配字符串开头。	\Aabc	abc	
\Z	仅匹配字符串末尾。	abc\Z	abc	
\b	匹配\w和\W之间。	a\b!bc	a!bc	
\B	[^\b]	a\Bbc	abc	

	代表左右表达式任意匹配一个。 它总是先尝试匹配左边的表达式，一旦成功匹配则跳过匹配右边的表达式。 如果 没有被包括在()中，则它的范围是整个正则表达式。	abc def	abc def
(...)	被括起来的表达式将作为分组，从表达式左边开始每遇到一个分组的左括号'('，编号+1。 另外，分组表达式作为一个整体，可以后接数量词。表达式中的 仅在该组中有效。	(abc){2} a(123 456)c	abcabc a456c
(?P<name>...)	分组，除了原有的编号外再指定一个额外的别名。	(?P<id>abc){2}	abcabc
\<number>	引用编号为<number>的分组匹配到的字符串。	(\d)abc\1	1abc1 5abc5
(?P=name)	引用别名为<name>的分组匹配到的字符串。	(?P<id>\d)abc(?P=id)	1abc1 5abc5
特殊构造（不作为分组）			
(?:...)	(...)的不分组版本，用于使用' '或后接数量词。	(?:abc){2}	abcabc
(?iLmsux)	iLmsux的每个字符代表一个匹配模式，只能用在正则表达式的开头，可选多个。匹配模式将在下文中介绍。	(?i)abc	AbC
(?#...)	#后面的内容将作为注释被忽略。	abc(?#comment)123	abc123
(?=...)	之后的字符串内容需要匹配表达式才能成功匹配。 不消耗字符串内容。	a(?=\d)	后面是数字的a
(?!...)	之后的字符串内容需要不匹配表达式才能成功匹配。 不消耗字符串内容。	a(?:!\d)	后面不是数字的a
(?<=...)	之前的字符串内容需要匹配表达式才能成功匹配。 不消耗字符串内容。	(?<=\d)a	前面是数字的a
(?<!...)	之前的字符串内容需要不匹配表达式才能成功匹配。 不消耗字符串内容。	(?<!\d)a	前面不是数字的a
(?(id/name) yes-pattern no-pattern)	如果编号为id/别名为name的组匹配到字符，则需要匹配yes-pattern，否则需要匹配no-pattern。 no-pattern可以省略。	(\d)abc(?:1)\d abc) http://www.cnblogs.com/huxi	1abc2 abcabc

python re库里的方法可以看这个[文章](#)。

常用正则表达式

一、校验数字的表达式

- 数字: ^[0-9]*\$
 - n位的数字: ^\d{[n]}\$
 - 至少n位的数字: ^\d{[n],}\$
 - m-n位的数字: ^\d{[m,n]}\$
 - 零和零点开头的数字: ^([0-9][0-9])*\$
 - 非零开头的最多带两位小数的数字: ^([1-9][0-9])*([0-9]{1,2})?*\$
 - 带1-2位小数的正数或负数: ^([-]?(\d+(\.\d{1,2}))?)\$
 - 正数、负数、和小数: ^([-]?(\d+(\.\d{1,2}))?)?*\$
 - 有两位小数的正实数: ^([0-9]{1,2})?([0-9]{2})?7\$
 - 有1-3位小数的正实数: ^([0-9]{1,2})?([0-9]{0,2})?7\$
 - 非零的正整数: ^([1-9]\d*)\$ 或 ^([1-9][0-9]*)((\.\d))\$ 或 ^+?([1-9][0-9])*\$
 - 非零的带小数: ^([1-9]\d*)?0*\$ 或 ^([1-9]\d*)?7\$
 - 非负整数: ^\d+\$ 或 ^([1-9]\d*)?0\$
 - 非正整数: ^[-1]\d+\$ 或 ^(([-1]\d*))?0\$
 - 非负浮点数: ^\d+(\.\d*)?7\$ 或 ^([1-9]\d+\.\d*)?0\$
 - 非正浮点数: ^([-1]\d+(\.\d*))?0\$ 或 ^([-1]\d+\.\d*)?0\$
 - 正浮点数: ^([1-9]\d+(\.\d*)?0\$)或^([1-9]\d+(\.\d*)?0\$)?0.0+0\$
 - 负浮点数: ^([-1]\d+(\.\d*)?0\$)或^([-1]\d+(\.\d*)?0\$)?0.0-0\$
 - 浮点数: ^([-1]\d+(\.\d*)?0\$)或^([-1]\d+(\.\d*)?0\$)?0.0+0\$或^([-1]\d+(\.\d*)?0\$)?0.0-0\$

校验字符的表达式

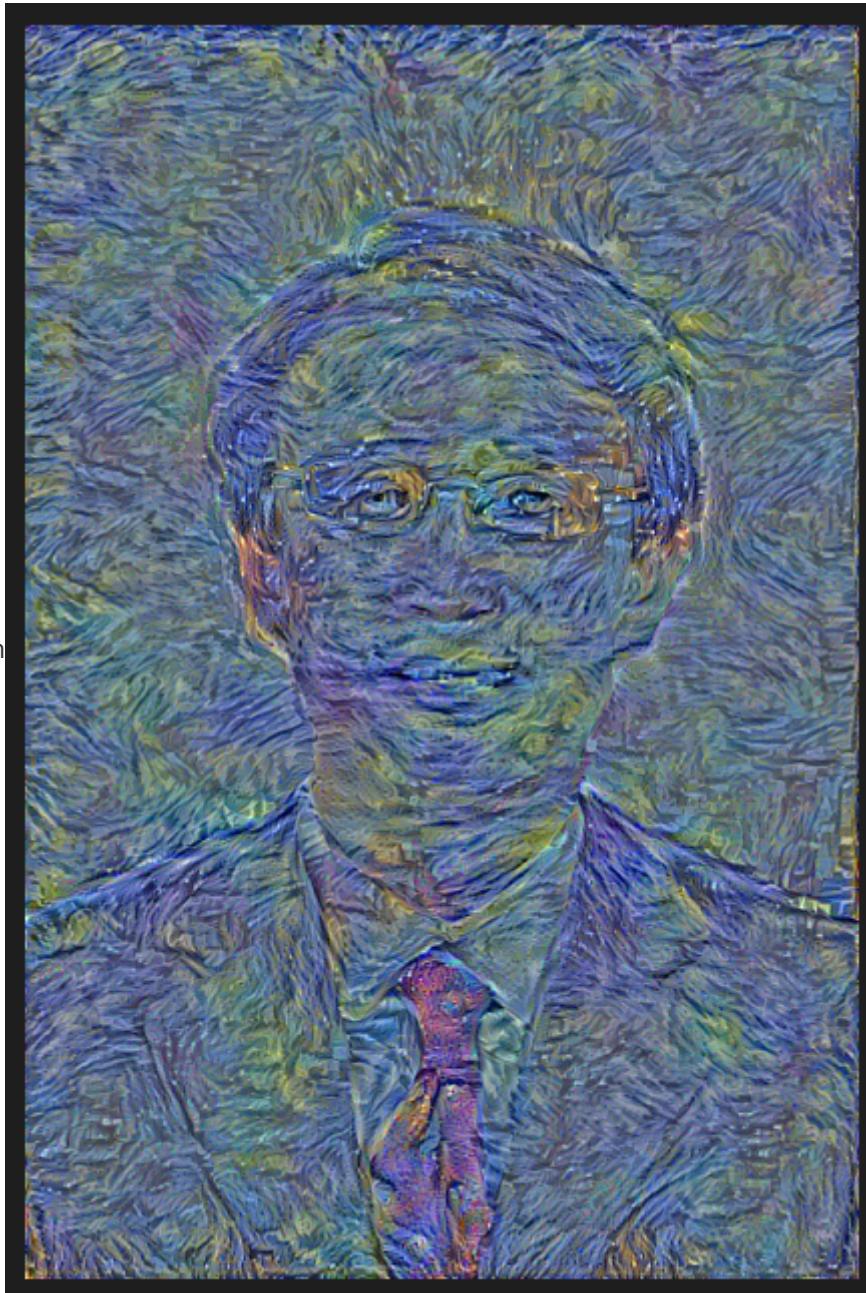
- 汉字：^\u4e00\u95f4[\u54cd\u54cd]\\$
 - 英文和数字：^|[A-Za-z0-9]+|s 或 ^[A-Za-z0-9]{4,40}\\$
 - 长度为3-20的所有字符：^|[3,20]\\$
 - 由26个英文字母组成的字符串：^|[A-Za-z]\\$
 - 由26个大写英文字母组成的字符串：^|[A-Z]\\$
 - 由26个小写英文字母组成的字符串：^|[a-z]\\$
 - 由数字和26个英文字母组成的字符串：^|[0-9A-Za-z]\\$
 - 由数字、26个英文字母或者下划线组成的字符串：^|w+w\\$|s 或 ^|w{3,20}\\$
 - 中文、英文、数字包括下划线：^|[u4e00-\u9fa5A-Za-z_0-9]+\\$
 - 中文、英文、数字但不包括下划线等符号：^|[u4e00-\u9fa5A-Za-z_0-9]*\\$ 或 ^|[u4e00-\u9fa5A-Za-z_0-9]{2,20}\\$
 - 可以输入含有“%”、“?”等字符：^|[%?]*\\$|x22+
 - 禁止输入含有的字符：^|[~]+\$

三、特殊需求表达式



2

1000epoch

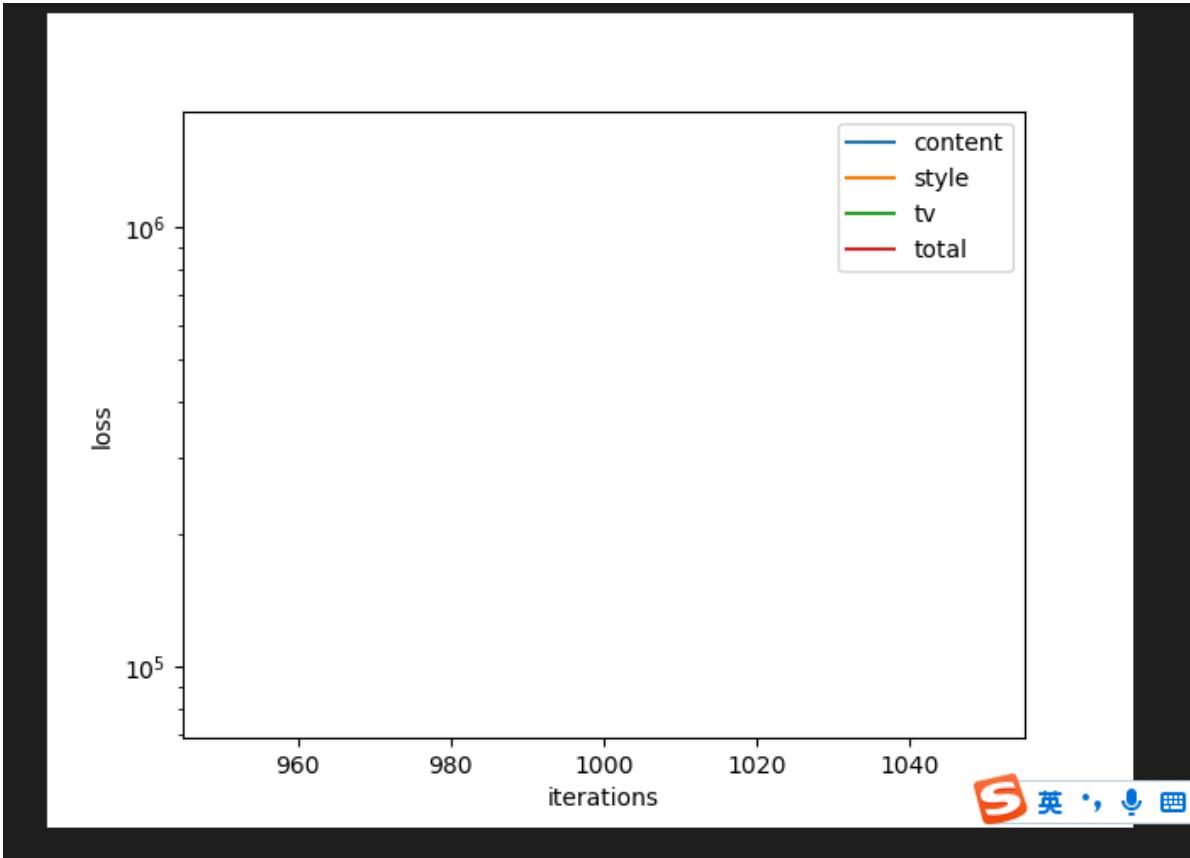


100epoch

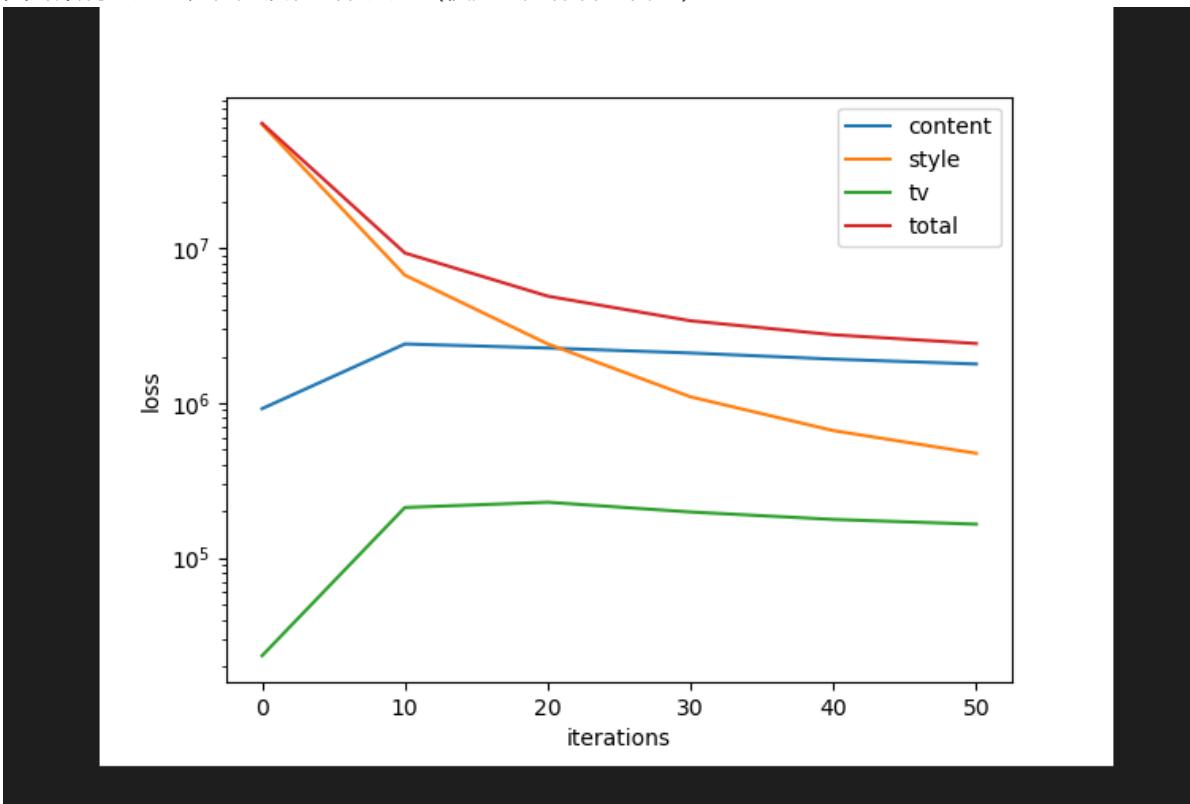


5kePOCH; 可以看到基本上1kePOCH就可以满足人眼审美需求了；5k则是更为精细的还原了内容细节
(跑了十六分钟。这个方法确实耗时)

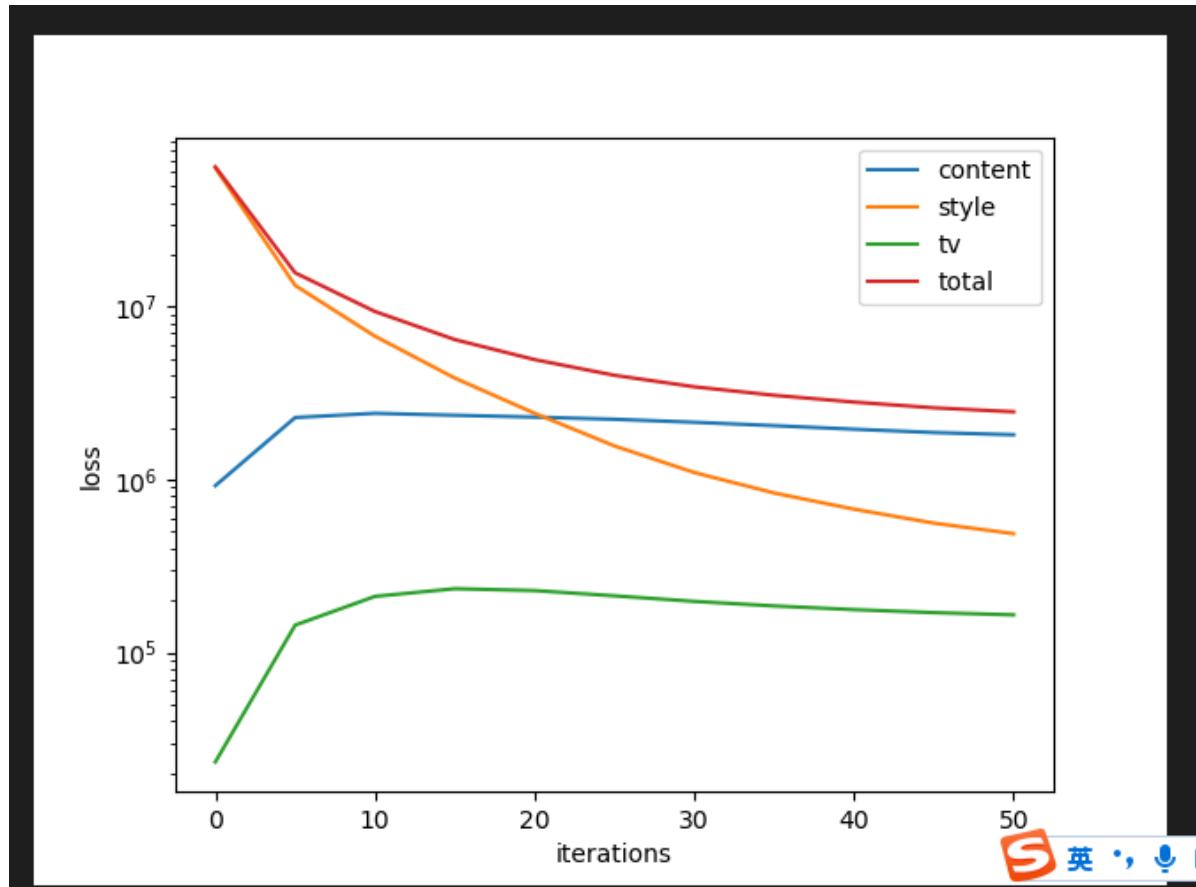
然后自己试图将loss画出来研究loss收敛在第几轮；出现了一点小问题



图片顺利生成了，但是没有绘制点。 (使用的是作者的代码)



解决了。要综合使用多个参数。那个参数--print-iterations 控制图中的颗粒度。上图是10，下图5



3 论文 (StackGan 提出了ca ; StackGan++ 提出了color-consistency regularization ; AttnGan)

3.1 StackGan:

Background

Task(what to do):

Synthesizing high-quality images from text descriptions 通俗来说，就是一个把文本描述转换为图像 (text-to-image) 的过程,问题涉及计算机视觉、自然语言处理的领域。

Challenges:

a.existing text to image approaches can roughly reflect the meaning of the given descriptions and fail to contain necessary details and vivid object parts.也就是说，现有的文本-图像方法生成的样本仅仅只可以粗略地反映给定描述的含义，但是往往不包含必要的细节和物体生动的部分，**细节缺失是重要的问题之一**。

b. it is very difficult to train GAN to generate high-resolution photo-realistic images from text descriptions.对于现有模型GAN来说，要实现高分辨率并不容易，首先，简单的在GAN模型中添加更多的上采样层用于生成高分辨率图像（例如，256×256）的方法通常导致**训练不稳定并产生无意义的输出**，如下图所示：

This bird is white with some black on its head and wings,	This bird has a yellow belly and tarsus, grey back, wings, and brown	This flower has overlapping pink pointed petals surrounding a ring
---	--	--



除此，GAN生成高维图片的主要问题在于，自然图像分布与模型分布在高维空间上几乎不交叠。当要生成的图像分辨率增大时，该问题更加明显。

Some Methods:

- a. 变分自动编码器 (VAE) 提出了以最大化数据似然的下限为目标的概率图模型。
- b. 利用神经网络模拟像素空间的条件分布的自回归模型 (例如PixelRNN)
- c. 生成对抗网络 (GAN) [已经显示出有希望生成清晰图像的性能。但是训练的不稳定性使得GAN模型很难生成高分辨率图像,基于能量的 GAN(energy-based GAN)也被提出用于更稳定的训练行为.]
- c. 超分辨率方法 (super-resolution methods) 只能为低分辨率图像添加有限的细节。
ansimov等通过学习估计文本和生成图像之间的对应的AlignDRAW模型。Reed等使用条件 PixelCNN 来使用文本描述和对象位置约束来生成图像。Nguyen等人使用近似 Langevin 抽样的方法来生成。
- d.....具体可见译文，这里不做具体展开。

Work

Do(How to do?)

- (1) We propose a novel Stacked Generative Adversarial Networks for synthesizing photo-realistic images from text descriptions. It decomposes the difficult problem of generating high-resolution images into more manageable subproblems and significantly improve the state of the art. The StackGAN for the first time generates images of 256×256 resolution with photo-realistic details from text descriptions.
- (2) A new Conditioning Augmentation technique is proposed to stabilize the conditional GAN training and also improves the diversity of the generated samples.
- (3) Extensive qualitative and quantitative experiments demonstrate the effectiveness of the overall model design as well as the effects of individual components, which provide useful information for designing future conditional GAN models.

简单来说即：

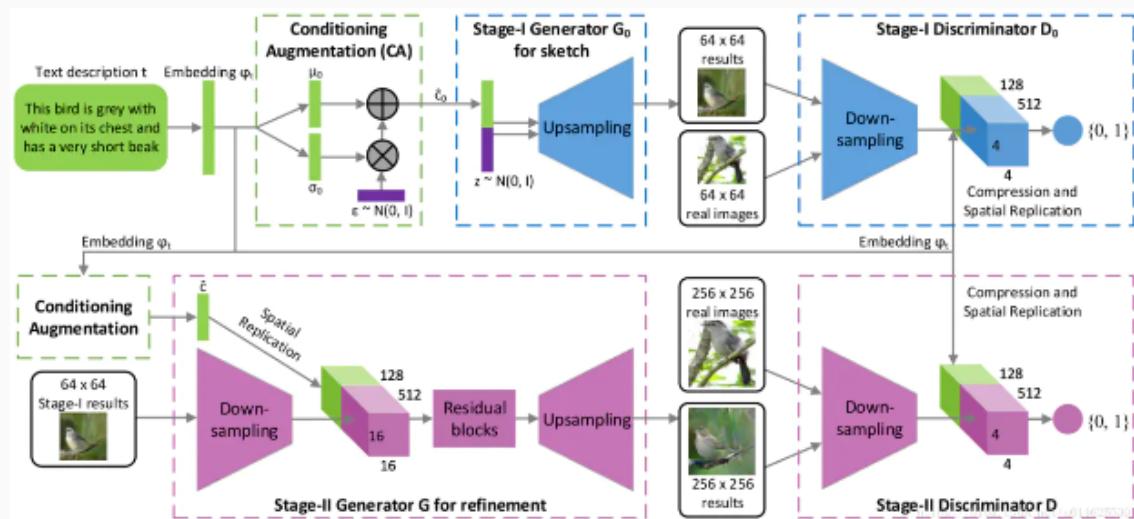
1. 提出了一种新的叠加生成对抗网络 (stackgan)，用于从文本描述中合成照片般逼真的图像。它将生成高分辨率图像的困难问题分解为更多可管理的子问题-----解决细节缺失

问题，分辨率变高。

2. 提出了一种新的条件增强技术 (CA) 来稳定条件GAN的训练，同时也提高了条件GAN训练的多样性。
3. 广泛的定性和定量实验证明整体模型设计的有效及单个组件的效果。

Model

接下来详细看一下模型，如下图



整个模型分为两个阶段：

- a.stack-I 第一阶段：勾画出在给定的文字描述上条件化的对象的原始形状和基本颜色，并从随机噪声向量中绘制背景布局，产生低分辨率图像。
- b.stack-II 第二阶段：修正了第一阶段的低分辨率图像中的缺陷，并通过再次读取文字描述来完成对象的细节，从而产生高分辨率的照片般逼真的图像。

现在先讲述一下条件增强技术 (CA) 再接着更好的详解第一二阶段。

条件增强技术 (CA)

Why? :

对于 text-to-image 生成任务，text-image 训练数据对数量有限，这将导致文本条件多样性的稀疏性(sparsity in the text conditioning manifold)，而这种稀疏性使得 GAN 很难训练。因此，我们提出了一种新颖的条件增强技术(Conditioning Augmentation technique)来改善生成图像的多样性，并稳定 conditional-GAN 的训练过程，这使得隐含条件分布更加平滑 (encourages smoothness in the latent conditioning manifold)。

How?

首先，文本描述 t 首先由编码器编码，产生文本 embedding φ_t 。

文本嵌入是非线性的转化为潜在条件变量作为生成器的输入。但是，文字的潜在空间的 embedding 通常是高维的 (> 100 维)，在数据量有限的情况下通常会导致潜在的数据流形的不连续，这是不可取的生成模型学习方法。为了缓解这个问题，作者引入条件增强

技术来产生额外的条件变量 \hat{c} , 这边的条件变量不是固定的而是作者从独立的高斯分布 $N(\mu(\varphi_t), \Sigma(\varphi_t))$ 随机地采样的。其中平均值 $\mu(\varphi_t)$ 和对角协方差矩阵 $\Sigma(\varphi_t)$ 是文本embedding φ_t 的函数。

条件增强在少量图像-文本对的情况下产生更多的训练数据，并有助于对条件流形的小扰动具有鲁棒性。为了进一步强化条件流形的平滑性并避免过拟合，作者在训练过程中将以下正则化项添加到生成器的目标中：

$$D_{KL}(N(\mu(\varphi_t), \Sigma(\varphi_t))||N(0, I))$$

这是标准高斯分布和条件高斯分布之间的LK散度。在条件增强中引入的随机性有利于对文本进行图像翻译建模，因为相同的句子通常对应于具有各种姿势和外观的对象。

Stack-I 第一阶段详解：

第一层GAN被用来简化任务，以便首先生成低分辨率的图像，这样的工作重点仅在于绘制粗糙的形状并为对象赋予正确的颜色。

过程详解：文本描述首先由一个预先训练的编码器，产生embedding φ_t ，文本embedding的高斯调节变量 \hat{c}_0 来自 $N(\mu_0(\varphi_t), \Sigma_0(\varphi_t))$ ，使得文本描述有多样性。第一层GAN以 \hat{c}_0 和随机变量 z 为条件，通过交替地最大化 L_{D_0} 和最小化 L_{G_0} ，来训练判别器 D_0 和生成器 G_0 。

$$\mathcal{L}_{D_0} = \mathbb{E}_{(I_0, t) \sim p_{data}} [\log D_0(I_0, \varphi_t)] + \mathbb{E}_{z \sim p_z, t \sim p_{data}} [\log(1 - D_0(G_0(z, \hat{c}_0), \varphi_t))], \quad (3)$$

$$\mathcal{L}_{G_0} = \mathbb{E}_{z \sim p_z, t \sim p_{data}} [\log(1 - D_0(G_0(z, \hat{c}_0), \varphi_t))] + \lambda D_{KL}(N(\mu(\varphi_t), \Sigma(\varphi_t))||N(0, I)). \quad (4)$$

其中真实图像 I_0 和文本描述 t 来自真实数据分布 p_{data} z 是从给定分布 p_z （本文中是高斯分布）随机采样的噪声向量。 λ 是一个正则化参数，可以平衡方程 (4) 中的两项。此处设定 $\lambda=1$ 。 $\mu_0(\varphi_t)$ 和 $\Sigma_0(\varphi_t)$ 都是从神经网络的其余部分学习得到的。

对于生成器 G_0 ，为了获得文本调节变量 \hat{c}_0 文本的embedding φ_t 是首先被馈送到完全连接的层中以产生高斯分布的 μ_0 和 σ_0 (σ_0 是 Σ_0 的对角线中的值)。 \hat{c}_0 然后从高斯分布中取样。 N_g 维条件矢量 \hat{c}_0 通过 $c_0 = \mu_0 + \sigma_0 \odot \epsilon$ 计算（其中 \odot 是元素乘法， $\epsilon \sim N(0, 1)$ ）。然后， \hat{c}_0 与一个 N_z 维噪声矢量拼接，通过一系列的上采样块生成一副的图像。

对于判别器 D_0 ，文本的embedding φ_t ，首先被全连接层压缩到 N_g 尺寸，随后进行空间性重复形成一个 $M_d \times M_d \times N_d$ 的张量。同时，图像通过一系列的下采样块，直到它具有 $M_d \times M_d$ 空间维度。然后，图像滤波器沿着信道维度与文本张量拼接。由此产生的张量进一步馈送到 1×1 的卷积层以在图像和文本上共同学习特征。最后，使用一个节点的全连接层来产生决策分数。

stack-II 第二阶段详解：

Stage-I 阶段生成的低分辨率图像通常缺乏鲜明的目标特征，并且可能包含一些变形。同时，文本描述中的部分信息可能也未体现出来。所以，通过 Stage-II 可以在 Stage-I 生成

的低分辨率图像和文本描述的基础上，生成高分辨率图片，其修正了 Stage-I 的缺陷，并完善了被忽略的文本信息细节。

以低分辨率的结果 $s_0 = G_0(z, \hat{c}_0)$ 和高斯分布的隐变量 \hat{c} 为条件，生成器 G 和判别器 D 在第二层GAN中被训练以最大化 L_D 和最小化 L_G

$$\begin{aligned}\mathcal{L}_D &= \mathbb{E}_{(I,t) \sim p_{data}} [\log D(I, \varphi_t)] + \mathbb{E}_{s_0 \sim p_{G_0}, t \sim p_{data}} [\log (1 - D(G(s_0, \hat{c}), \varphi_t))], \\ \mathcal{L}_G &= \mathbb{E}_{s_0 \sim p_z, t \sim p_{data}} [\log (1 - D(G(s_0, \hat{c}), \varphi_t))] + \lambda D_{KL}(\mathcal{N}(\mu(\varphi_t), \Sigma(\varphi_t)) | \mathcal{N}(0, I)).\end{aligned}$$

与原来的GAN公式不同的是，随机噪声 z 在这个阶段没有被使用，而是假设随机性已经被 s_0 保存了。在这个阶段使用的高斯条件变量 \hat{c} 和在第一阶段GAN使用的 \hat{c}_0 共享相同的预先训练的文本编码器，生成相同的文本嵌入 φ_t 。但第一层和第二层的条件增强不同的全连接层，用于产生不同的均值和标准偏差。通过这种方式，第二层GAN学习捕获第一层GAN所忽略的文本embedding中的有用信息。

第二层生成器是一个带有残余块的编码器-解码器网络。类似于前一阶段，文本的 embedding φ_t 被用来生成 N_g 维空间的文本调节向量 \hat{c} 这是在空间中重复所形成的一个 $M_g \times M_g \times N_g$ 的张量。同时，第一层GAN产生的结果被送入几个下采样块（即编码器），直到它具有 $Mg \times Mg$ 的空间尺寸。图像特征和文本特征沿着通道的维度被拼接。编码的图像特征加上文本特征被馈送到多个残差块，其被设计成学习跨图像和文本特征的多模式表示。最后是一系列的上采样层（即解码器）被用来生成一个高分辨率图片。这种生成器能够帮助纠正输入图像的缺陷，同时添加更多细节来生成逼真的高分辨率图像。

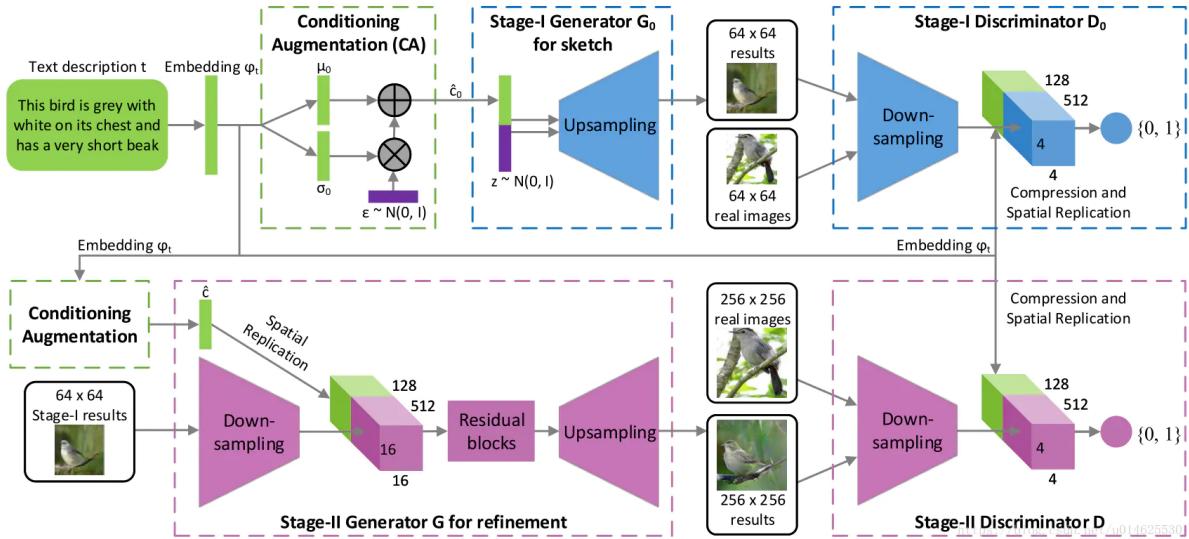
对于判别器而言，其结构与仅具有额外的下采样块第一阶段判别器的结构类似，因为在此阶段图像尺寸较大。为了明确强制GAN学习图像和条件文本之间的更好的对应，作者采用了Reed等人提出的匹配感知判别器 在训练期间，判别器将真实图像及其对应的文本描述作为正样本对，而负样本对包括两组：首先是真实的图像与不匹配的文本嵌入，而第二个是合成图像与相应的文本的embedding。

Experiment

进入实验部分

Detail:

上采样网络由 nearest-neighbor 上采样组成。上采样后，接一个 3x3，步长为 1 的卷积



评价指标

Inception Score

$$I = \exp(E_x D_{KL}(p(y | x) || p(y)))$$

其中 x 表示一个生成的样本， y 是由 Inception 模型预测的标签]。这个指标背后的含义是好的模型应该产生多样但有意义的图像。因此，边际分布 $p(y)$ 和条件分布 $p(y|x)$ 之间的 KL 散度应该很大。（为什么会很大？主要是二者分布的分布原因。详细解析见：[全面解析 Inception Score 原理及其局限性](#)）

The bird is completely red → The bird is completely yellow



This bird is completely red with black wings and pointy beak → this small blue bird has a short pointy beak and brown on its wings



句子embedding的插值:为了进一步证明StackGAN学习了一个平滑的潜在数据流形，作者使用它来从线性内插的句子embedding中生成图像，如上图所示。我们令噪声矢量为 z ，生成的图像是从给定的文字描述中推断出来的。第一行的图像是通过制作的简单句子来合成的。这些句子只包含简单的颜色描述。结果表明，从插值嵌入生成的图像可以准确地反映颜色的变化，并生成似是而非的鸟的形状。第二行说明从更复杂的句子中产生的样本，其中包含更多关于鸟类出现的细节。生成的图像将其主要颜色从红色更改为蓝色，并将翼色从黑色更改为棕色。

Conclusion

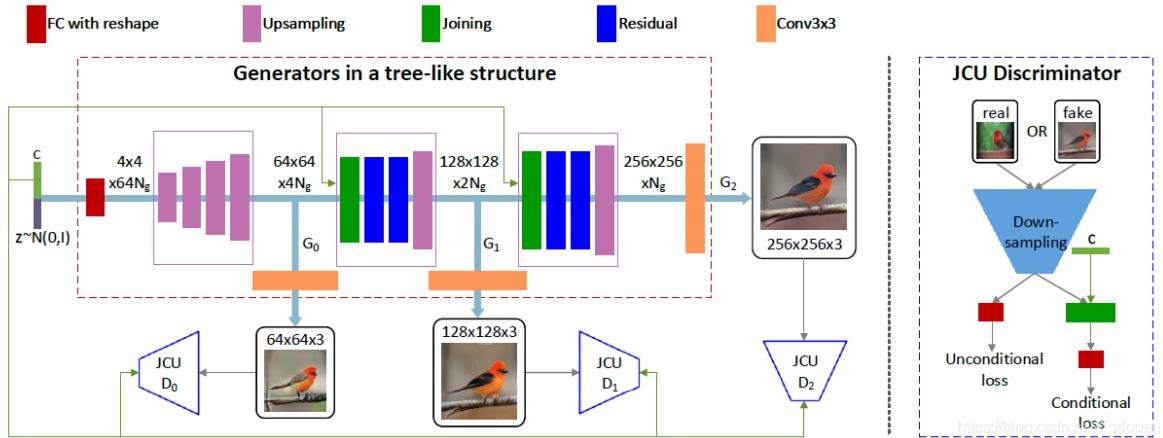
本文提出了 Stacking Generative Adversarial Networks (StackGAN)，其中使用了条件增强来合成相片逼真的图像。所提出的方法将文本到图像合成分解成两步：草图-细化过程。第一阶段的GAN根据给定文字描述的基本颜色和形状约束绘制对象。第二阶段的GAN纠正了第一阶段结果中的缺陷并增加了更多细节，从而产生更高分辨率的图像并获得更好的图像质量。广泛的定量和定性结果证明了所提出的方法的有效性。与现有的文字到图片生成模型相比，StackGAN生成更高分辨率的图像（如 256×256 ），并具有更逼真的细节和多样性。

一点启发：把大的问题进行分解多部来完成，加以条件逐步细化。

3.2 StackGan++

本文认为StackGAN++的优势主要是以下三点：

虽然仍然是采用多阶段逐级提高生成图像的分辨率的方式，但是不同于之前的两阶段分开训练，
 StackGAN++可以采用end-to-end的方式进行训练
 对于无条件图像生成提出了一种新的正则化方式color-consistency regularization来帮助在不同的分辨率下生成更一致的图像
 既可以用于text-to-image这样的条件图像生成任务，也可以用于更一般的无条件图像生成任务，均可以取得比其他模型更优异的结果
 模型整体的架构如下所示：



第一点

从上图可以看出，StackGAN++生成图像的分辨率变化是 $4 \times 4 \rightarrow 64 \times 64 \rightarrow 128 \times 128 \rightarrow 256 \times 256$ ，整体上构成一种树性的结构。如果将每个阶段的部分称为一个分支，那么在每个分支下生成图像用于捕获该分辨率下图像的分布，判别器用于估计生成图像是来自同分辨率真实数据分布而非生成器的概率。之所以可以采用这样的结构帮助生成高分辨率的图像，原理和我之前在StackGAN中提高的思想是一致的。将两阶段的生成变为四阶段的生成方式，那么分辨率越低，生成图像所满足的分布和真实图像所满足的分布之间的交叠就越多，生成器的训练就越容易。

假设随机噪声向量 $z \sim p_{noise}$ 采样自标准正态分布，将其在多个隐藏层之间做非线性变换得到 h_i 作为 G_i 的输入，转换方式为

$$h_0 = F_0(z); \quad h_i = F_i(h_{i-1}, z), i = 1, 2, \dots, m-1$$

其中 h_i 是第*i*个分支隐藏层的特征，*m*是分支的总数， F_i 通常是神经网络。这里在每个 h_i 的产生过程中都会接收 z 做为输入，作者认为这样的方式可以捕获前面分支忽略的信息。

在得到了关于 h_i 的结果序列 $(h_0, h_1, \dots, h_{m-1})$ 后，使用 $s_i = G_i(h_i), i = 0, 1, \dots, m-1$ 就可以生成分辨率从低到高的图像。

目标函数这里并没有太大变化，仍然采用交叉熵 $\mathcal{L}_{D_i} = -\mathbb{E}_{x_i \sim p_{data_i}} [\log D_i(x_i)] - \mathbb{E}_{s_i \sim p_{G_i}} [\log (1 - D_i(s_i))]$ 。这样不同分辨率下的判别器只需关注某个分辨率尺度即可，另外它们可并行训练。生成器在判别器固定的情况下，不断的逼近自己所关注的分辨率下图像所满足的真实分布，目标函数为 $\mathcal{L}_G = \sum_{i=1}^m \mathcal{L}_{G_i}, \quad \mathcal{L}_{G_i} = -\mathbb{E}_{s_i \sim p_{G_i}} [\log D_i(s_i)]$ 。

通过这样的方式，StackGAN++就实现了使用端到端的方式最后生成 256×256 的高分辨率图像。

第二点

因为在整个图像的生成过程中，不同分辨率图像可能细节部分会有不同，但是它们在结构和颜色的基本方面应是一致的。因此可以通过在这些方面加些限制，帮助生成器生成色彩一致的高质量图像。

将生成图像所有的像素表示为 $\mathbf{x}_k = (R, G, B)^T$ ，然后得到 μ 和 Σ ，计算公式为

$$\mu = \sum_k \mathbf{x}_k / N, \quad \Sigma = \sum_k (\mathbf{x}_k - \mu)(\mathbf{x}_k - \mu)^T / N$$

而本文所提出的color-consistency regularization所做的就是最小化不同分辨率图像下 μ 和 Σ 的差别，目标函数为

$$\mathcal{L}_{C_i} = \frac{1}{n} \sum_{j=1}^n \left(\lambda_1 \left\| \boldsymbol{\mu}_{s_i^j} - \boldsymbol{\mu}_{s_{i-1}^j} \right\|_2^2 + \lambda_2 \left\| \boldsymbol{\Sigma}_{s_i^j} - \boldsymbol{\Sigma}_{s_{i-1}^j} \right\|_F^2 \right)$$

但它只用在无条件图像生成中，因为在条件图像生成任务中，所附件的条件已经提供了一种很强的限制，这种低尺度下的限制就没有那么重要了。

第三点

对于无条件图像生成来说，判别只需要关注图像是真还是假，但是对于条件图像生成来说，除了判别真假外，还需要判别所给的图像是否和给的条件相符。因此关于 h_i 的计算式为 $h_0 = F_0(c, z), \quad h_i = F_i(h_{i-1}, c)$ ，这里使用 c 而不是 z 是希望生成器更多的关注所给条件中所包含的信息。生成器和判别器的目标函数为

$$\begin{aligned} \mathcal{L}_{D_i} &= \underbrace{-\mathbb{E}_{x_i \sim p_{data_i}} [\log D_i(x_i)] - \mathbb{E}_{s_i \sim p_{G_i}} [\log (1 - D_i(s_i))]}_{\text{unconditional loss}} + \underbrace{-\mathbb{E}_{x_i \sim p_{data_i}} [\log D_i(x_i, c)] - \mathbb{E}_{s_i \sim p_{G_i}} [\log (1 - D_i(s_i, c))]}_{\text{conditional loss}} \\ \mathcal{L}_{G_i} &= \underbrace{-\mathbb{E}_{s_i \sim p_{G_i}} [\log D_i(s_i)]}_{\text{unconditional loss}} + \underbrace{-\mathbb{E}_{s_i \sim p_{G_i}} [\log D_i(s_i, c)]}_{\text{conditional loss}} \end{aligned}$$

但因为生成器使用了color-consistency 正则项，所以最后的目标函数为 $\mathcal{L}'_{G_i} = \mathcal{L}_{G_i} + \alpha * \mathcal{L}_{C_i}$ 。

总结

整体上来说，StackGAN++做为StackGAN的改进版，它可以以端到端的方式生成更好质量的图像，而且训练过程更加稳定。但是StackGAN分阶段的训练方式收敛速度更加快，所需的GPU内存也更少。

3.3 AttnGan

在本文中，我们提出了一种注意生成对抗网络（AttnGAN），它允许由注意力驱动的、多阶段的细化来进行细粒度的文本到图像的生成。通过一种新的注意生成网络，AttnGAN可以通过关注自然语言描述中的相关词，在图像的不同子区域合成细粒度的细节。此外，还提出了一种深度注意多模态相似度模型来计算一个细粒度的图像-文本匹配损失来训练生成器。提出的AttnGAN显著优于之前的技术水平，在CUB数据集上的最佳报告初始分数提高了14.14%，在更具挑战性的COCO数据集上提高了170.25%。我们还通过可视化AttnGAN的注意层来进行详细的分析。这首次表明，分层注意GAN能够在单词级自动选择条件来生成图像的不同部分。

本文是StackGan++ (StackGAN++:Realistic Image Synthesis with Stacked Generative Adversarial Networks) 的后续工作

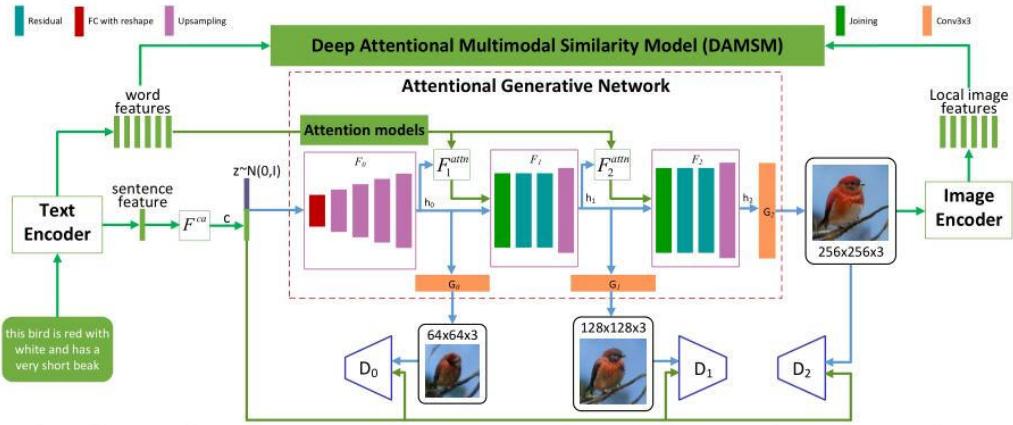


Figure 2. The architecture of the proposed AttnGAN. Each attention model automatically retrieves the conditions (*i.e.*, the most relevant word vectors) for generating different sub-regions of the image; the DAMSM provides the fine-grained image-text matching loss for the generative network.
https://blog.csdn.net/sinat_29983957

在本文中作者提出了一个 Attentional Generative Adversarial Network(AttnGAN),一种attention-driven的多stage的细粒度文本到图像生成器。

并借助一个深层注意多模态相似模型(deep attentional multimodal similarity model)来训练该生成器。

它首次表明 the layered attentional GAN 能够自动选择单词级别的condition来生成图像的不同部分。

该模型由两部分组成

1. attentional generative network

该部分使用了注意力机制来生成图像中的子区域，并且在生成每个子区域时还考虑了文本中与该子区域最相关的词。如下图所示：



Figure 1. Example results of the proposed AttnGAN. The first row gives the low-to-high resolution images generated by G_0 , G_1 and G_2 of the AttnGAN; the second and third row shows the top-5 most attended words by F_1^{attn} and F_2^{attn} of the AttnGAN, respectively. Here, images of G_0 and G_1 are bilinearly upsampled to have the same size as that of G_2 for better visualization.²⁹⁹⁶³⁹⁵⁷

2. Deep Attentional Multimodal Similarity Model (DAMSM)

该部分用来计算生成的图像与文本的匹配程度。用来训练生成器。

Pipeline:

- 输入的文本通过一个Text Encoder 得到 sentence feature 和 word features
- 用sentence feature 生成一个低分辨率的图像 I_0
- 基于 I_0 加入 word features 和 sentence feature 生成更高分辨率细粒度的图像

Attentional Generative Adversarial Network

下面将分别介绍之前提到的两个模块。

Attentional Generative Network

从Figure 2 中可以看出该attentional generative network共有m个生成器(G_0, G_1, \dots, G_{m-1})它们的输入分别是(h_0, h_1, \dots, h_{m-1})生成($\hat{x}_0, \hat{x}_1, \dots, \hat{x}_{m-1}$)

其中

$$\begin{aligned} h_0 &= F_0(z, F^{ca}(\bar{e})); \\ h_i &= F_i(h_{i-1}, F_i^{attn}(e, h_{i-1})) \text{ for } i = 1, 2, \dots, m-1; \\ \hat{x}_i &= G_i(h_i). \end{aligned} \quad (1)$$

z 是noise,服从标准正态分布

\bar{e} 是global sentence vector

e 是word vector的矩阵

F^{ca} 是将 \bar{e} 转换为conditioning vector的Conditioning Augmentation 方法

F_i^{attn} 是第*i*个stage的attention model

$F^{attn}(e, h)$ 有两个输入, e 和 h

h 的每一列就是图像中一个子区域的feature vector

$\cdot F^{attn}(e, h) = (c_0, c_1, \dots, c_{N-1}) \in \mathbb{R}^{\tilde{D} \times \tilde{N}}$.

其中

$$c_j = \sum_{i=0}^{T-1} \beta_{j,i} e'_i, \text{ where } \beta_{j,i} = \frac{\exp(s'_{j,i})}{\sum_{k=0}^{T-1} \exp(s'_{j,k})}, \quad (2)$$

$$s'_{j,i} = h_j^T e'_i.$$

$\beta_{i,j}$ 表示在生成第*j*个子区域时attends到第*i*个word的权重。

生成器的目标函数定义如下:

$$\mathcal{L} = \mathcal{L}_G + \lambda \mathcal{L}_{DAMSM}, \text{ where } \mathcal{L}_G = \sum_{i=0}^{m-1} \mathcal{L}_{G_i}. \quad (3)$$

第*i*个stage的生成器 G_i 有对应的判别器 D_i , G_i 的对抗loss定义为:

$$\mathcal{L}_{G_i} = \underbrace{-\frac{1}{2} \mathbb{E}_{\hat{x}_i \sim p_{G_i}} [\log(D_i(\hat{x}_i))]}_{\text{unconditional loss}} - \underbrace{\frac{1}{2} \mathbb{E}_{\hat{x}_i \sim p_{G_i}} [\log(D_i(\hat{x}_i, \bar{e}))]}_{\text{conditional loss}}, \quad (4)$$

unconditional loss决定了img是real或者fake, conditional loss决定img和sentence是否匹配。

同理, 判别器的loss定义为

$$\begin{aligned} \mathcal{L}_{D_i} = & \underbrace{-\frac{1}{2} \mathbb{E}_{x_i \sim p_{data_i}} [\log D_i(x_i)] - \frac{1}{2} \mathbb{E}_{\hat{x}_i \sim p_{G_i}} [\log(1 - D_i(\hat{x}_i))]}_{\text{unconditional loss}} + \\ & \underbrace{-\frac{1}{2} \mathbb{E}_{x_i \sim p_{data_i}} [\log D_i(x_i, \bar{e})] - \frac{1}{2} \mathbb{E}_{\hat{x}_i \sim p_{G_i}} [\log(1 - D_i(\hat{x}_i, \bar{e}))]}_{\text{conditional loss}}, \end{aligned} \quad (5)$$

Deep Attentional Multimodal Similarity Model

DAMSM学习了两个神经网络(text encoder-LSTM,image encoder -CNN), 将图像的子区域和句子中的词映射到同一个语义空间来计算相似度, 在训练生成器的时候就可以通过计算img-text similarity得到一个fine-grained loss

The text encoder

文本编码器是一个双向LSTM网络, 用来提取文本描述(text description)的语义向量(semantic vectors)。

作者将该双向LSTM网络的hidden states认为是词的语义(semantic meaning).

所有词的特征矩阵为 e 纬度为 D , D 是 word vector的纬度, T 是词的数量。第*i*列 e_i 是第*i*个词的feature vector
并且, 最后一次hidden states被认为是 global sentence vector \bar{e} 同样是 D 纬的。

The image encoder

图像编码器是一个将图像映射到语义空间的CNN网络。中间层的输出被认为是图像不同子区域的局部特征, 后面层的输出是图像的全局特征。

作者从Inception-v3的mixed_6e层得到local feature $f(768 \times 289)$, 每一列是一个子区域的feature vector, 768是feature vector的纬度, 289是子区域的个数。

并且, global feature \bar{f} 的纬度是2048

然后将img feature映射到text feature同一个空间, 得到 v 和 \bar{v}

$$v = Wf, \quad \bar{v} = \bar{W}\bar{f}, \quad (6)$$

The attention-driven image-text matching score

该score是用来衡量img-sentence pair的匹配程度。

先计算句子中所有可能的单词对和图像中的子区域的相似度矩阵s。

$$s = e^T v$$

$s_{i,j}$ 是句子的第*i*个单词和图像的第*j*个子区域之间的点积相似度。

用如下方法归一化s

$$\bar{s}_{i,j} = \frac{\exp(s_{i,j})}{\sum_{k=0}^{T-1} \exp(s_{k,j})}. \quad (8)$$

然后, 作者构建了一个注意模型来计算每个词的region-context vector。 c_i 是与句子的第*i*个词相关的图像的子区域的动态表示。

$$c_i = \sum_{j=0}^{288} \alpha_j v_j, \quad \text{where } \alpha_j = \frac{\exp(\gamma_1 \bar{s}_{i,j})}{\sum_{k=0}^{288} \exp(\gamma_1 \bar{s}_{i,k})}. \quad (9)$$

γ_1 决定了在计算词的region-context vector时对其相关图像子区域特征的重视程度。

作者使用 c_i 和 e 之间的余弦相似度来定义第*i*个词和图像之间的相关性。

$$i.e., R(c_i, e_i) = (c_i^T e_i) / (\|c_i\| \|e_i\|).$$

最后整张图像(Q)和描述(D)的match score被定义为

$$R(Q, D) = \log \left(\sum_{i=1}^{T-1} \exp(\gamma_2 R(c_i, e_i)) \right)^{\frac{1}{\gamma_2}}, \quad (10)$$

DAMSM Loss

对一个batch的img-sentence 对 $\{(Q_i, D_i)\}_{i=1}^M$,

D_i 与 Q_i 匹配的后验概率被为

$$P(D_i|Q_i) = \frac{\exp(\gamma_3 R(Q_i, D_i))}{\sum_{j=1}^M \exp(\gamma_3 R(Q_i, D_j))}, \quad (11)$$

loss使用负对数后验概率(negative log posterior probability)

$$\mathcal{L}_1^w = - \sum_{i=1}^M \log P(D_i|Q_i), \quad (12)$$

w for word

$$\mathcal{L}_2^w = - \sum_{i=1}^M \log P(Q_i|D_i), \quad (13)$$

若将公式10重新新定义为

$R(Q, D) = (\bar{v}^T \bar{e}) / (\|\bar{v}\| \|\bar{e}\|)$, 再通过公式11和12就能得到

L_1^s 和 L_2^s s for sentence

于是DAMSM loss 为:

$$\mathcal{L}_{DAMSM} = \mathcal{L}_1^w + \mathcal{L}_2^w + \mathcal{L}_1^s + \mathcal{L}_2^s. \quad (14)$$

结果



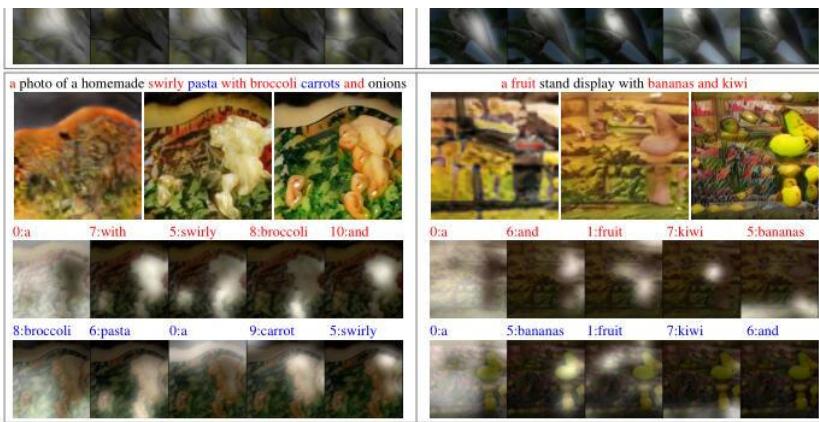


Figure 4. Intermediate results of our AttnGAN on CUB (top) and COCO (bottom) test sets. In each block, the first row gives 64×64 images by G_0 , 128×128 images by G_1 and 256×256 images by G_2 of the AttnGAN; the second and third row shows the top-5 most attended words by F_1^{attn} and F_2^{attn} of the AttnGAN, respectively. Refer to the supplementary material for more examples. [elshafai_29963957](#)



Figure 5. Example results of our AttnGAN model trained on CUB while changing some most attended words in the text descriptions.



Figure 6. 256×256 images generated from descriptions of novel scenarios using the AttnGAN model trained on COCO. (Intermediate results are given in the supplementary material.)



Figure 7. Novel images by our AttnGAN on the CUB test set.

论文地址[here](#)

代码地址[here](#)

最后