第 5 章

CHAPTER 5

文本转图像实验

在浩如烟海的诗词歌赋里，有多少使你身临其境，难以忘怀的名句呢？

它可以是韦应物的“独怜幽草涧边生，上有黄鹂深树鸣”；可以是李益的“一鸟白如雪，飞向白楼前”；也可以是唐李郢的“越鸟青春好颜色，晴轩入户看帖衣”。

但在你细细品味，沉醉诗词中之时，却发现脑海中的那只“鸟”是那么虚无缥缈、难以形容。此时，或许你将感慨，若在那个时代已经有了照相机，那该多好！诗人们就可以将所见所闻记录下来，配上自己的诗词，然而这一切也仅仅是感慨。

在这个相信科技的时代，科技将赐予你“超能力”，帮助你将感慨变为现实。当你陶醉于诗词佳句之中时，你的“超能力”将会帮助你“见”诗人所见，“闻”诗人所闻。如图 5.1 所示，韦应物的黄鹂、李益的白鸟、唐李郢的越鸟都将一一呈现在你的眼前，你的“超能力”将实现你想要的“身临其境”。

(a) 生成的黄鹂

(b) 生成的白鸟

(c) 生成的越鸟

图 5.1　诗句生成鸟类图片

5.1　背景介绍

科技所赐予你的“超能力”是什么呢？其实就是接下来所要介绍的文本转图像实验。那么，什么是文本转图像？这种技术又有什么用呢？

文本转图像就是当你输入一句话后，根据句意生成与其相符的图片。这种技术可以将诗词歌赋转化为“真实”的图像表达，以诗生画，这也是文本转图像的第一个应用场景。这项技术在肖像描述中也可以大展身手，当警察输入嫌疑人可能的外貌时，这项技术可以根据输入的外貌描述，帮助警察侧写出嫌疑人可能的长相以便追捕犯人。不仅如此，这项技术也可以应用到搜索引擎中，输入文本时，可以自动产生一系列符合描述的图片，用户可以自由选

择自己需要的图像。

在2016年以前，基本上所有的图像生成都是利用变分自编码器[1](Variational Auto-Encoder，VAE)和DRAW[2](Deep Recurrent Attentive Writer)方法来完成的，VAE基于贝叶斯公式以及相对熵的推导，利用最大似然估计生成图像，而DRAW通过在循环神经网络中加入注意力机制，每次生成一个关注的图像区域，接着将生成的局部图像进行叠加，得到生成的结果。而AlignDRAW[3]则可以在DRAW方法基础上加入文本对齐，使得图像生成更加精准。

2016年GAN-INT-CLS[4]出现以后，文本转图像的任务基本上都是使用GAN的方法完成的。GAN-INT-CLS是以对抗生成网络为主干模型，利用文本特征作为输入，在生成器中将文本特征与随机噪声拼接后输入生成器中，生成与文本匹配的图像，在判别器中对生成的假图像匹配了正确的文本以及真实图像但匹配了错误文本这两种错误进行分类，来辨别图像是否是生成器生成的。之后出现了Generative Adversarial What-Where Network[5](GAWWN)，它在原来的GAN的基础上添加了包围框和关键点的限定，达到生成更加精确的图像的效果。

StackGAN[6]使用了两个GAN来分步生成图像。它认为仅仅通过在网络中添加上采样的方法无法进一步提高生成图像的分辨率，所以StackGAN利用两个阶段的GAN来提高生成图像的精度，它首先利用图像的背景、颜色及轮廓等基本信息在第一阶段的GAN生成一幅低分辨率的图像，接着将生成的低分辨率的图像作为第二阶段的GAN网络输入，同时再次使用文本特征并对文本特征加入一些实用的随机噪声，使得生成的图像更加精确清楚。后来的StackGAN++[7]在原来的StackGAN的基础上改进，将GAN扩充成一个树状的结构，利用多个生成器和判别器同时进行训练，来获取多精度的生成图像。将低精度图像作为更高一级的生成器的输入方式，来达到生成更高精度图像的目的。该方法的好处是不仅可以完成限定性的生成任务，同时也扩展到了非限定性生成任务。

后续工作是在StackGAN++基础上提出来的AttnGAN[8]，也就是本实验所用的实验原理。AttnGAN在StackGAN++的基础上引入了注意力机制，不仅利用句子特征作为全局的约束，同时还利用得到的词特征，通过注意力机制来优化生成局部的图像，从而得到更为精确清晰的图像。此外，论文中还提出了一种DAMSM (Deep Attentional Multimodal Similarity Model)机制。它不仅仅考虑了原始的GAN损失，还在生成高精度图像后提取该图片的局部特征跟词嵌入进行对照获得DAMSM损失，使得模型训练更加关注文本细节的生成情况，从而达到提升生成效果的目的。

5.2 算法原理

本实验是基于2018年CVPR上的一篇名为*AttnGAN: Fine-Grained Text To Image Generation with Attentional Generative Adversarial Networks*的论文，文中提出了Generation with Attentional Generative Adversarial Networks(AttnGAN)模型，该模型是利用注意力机制以及GAN，生成在细节上更符合文本描述的图像。

AttnGAN模型生成图像的流程图的概括如图5.2所示。首先，需要将输入的句子利用文本编码器编码成为计算机能够识别处理的向量形式，得到词的向量形式和整个句子的向

量形式；其次，将句子向量输入神经网络 1 得到低分辨图像；然后，将全局图像和词向量输入注意力机制，得到每个词与低分辨图像的局部区域的相关性，相关性大则对应关系的权重值就大，否则就小；之后将权重和低分辨图像输入神经网络 2，利用权重对图像的局部进行分辨率改善；最后，经过 GAN 的生成器生成高分辨率的图像。

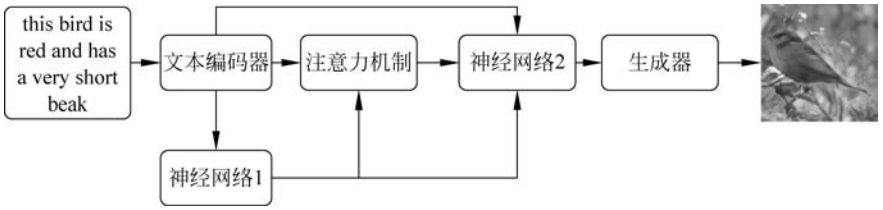


图 5.2　实验流程概括

5.2.1　词向量

词向量也可以叫作词嵌入[9](word embedding)，它的本质是用来表示自然语言中的词或字的一个多维向量。那它是怎么产生的呢？其实很简单，大家对字典应该很熟悉吧，在字典上，每一个字都按照字的拼音以及笔画的多少得到一个独一无二的位置。其实词向量也是相同的道理，词向量也有属于自己的“字典”。在“字典”中，只有有限数量的字词，那么人们就想到用一组固定长度二进制码来表示“字典”中的词向量。假设在“字典”中有 n 个词，那么这组固定长度二进制码的长度为 n，也就是说每个词对应的词向量的二进制码中只有一个数为 1，其余的数都为零。这种编码的方式叫作独热编码(one-hot encoding)。

但是这种方法没有考虑词与词之间的关系以及词向量维度过高的问题，word2vec[9]词向量解决了这个问题。word2vec 词向量的产生过程如图 5.3 所示。首先需要输入和这个词相关的 k 个词的独热编码 $\boldsymbol{w}^1_{1\times n}\cdots\boldsymbol{w}^k_{1\times n}$，经过权值映射 $\boldsymbol{w}^i_{1\times n}\boldsymbol{W}^i_{n\times m}$，求和均值处理 $\sum_{i=1}^{k}\boldsymbol{w}\boldsymbol{W}$ 就可以得到该词的 word2vec 词向量 $\boldsymbol{c}_{1\times m}$。将 word2vec 词向量输入隐藏层 2(参数大小和数量与隐藏层 1 都不同)和 Softmax 层可以得到每个词的概率分布 $w'_{1\times n}$，而这个分布概率最大值处和独热编码向量的“1”处位置应该是一致的，也正是如此，利用概率分布和独热编码得到损失函数，两个隐藏层的权重矩阵可以通过反向传播的方式训练得到。利用这样的方式，由于输入是与所求的向量相关的词，并且利用权重矩阵进行了低维映射，所以得到的

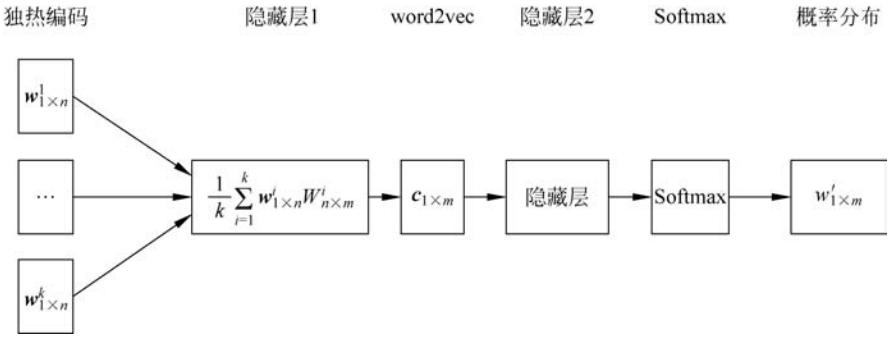


图 5.3　word2vec 词向量的生成流程图

word2vec 词向量的维度可以人为选定为较低的维度，并且包含了词与词之间的关系。这种词与词的关系可以由求两个向量的余弦距离得到，词的意思越相近，词向量的余弦距离就越小。

5.2.2 双向长短时记忆网络

这部分将简单介绍一下双向长短时记忆网络[10]（bi-directional Long Short Time Memory，bi-LSTM）。你可能会问，词向量已经可以被计算机理解了，bi-LSTM 要用来做什么？它的作用就是让计算机记住以前输入的词向量。

在句子中词按顺序输入网络时，有些词的意思和上下文有关，比如代词等，如果没有记住之前的词的话，只考虑当前词的话是无法被理解的，所以在 LSTM 中有细胞状态和隐藏状态（隐藏状态也代表特征）可以通过遗忘门、记忆门、输出门选择性记住以前输入的有用信息，遗忘无用信息并输出，其流程如图 5.4 所示。

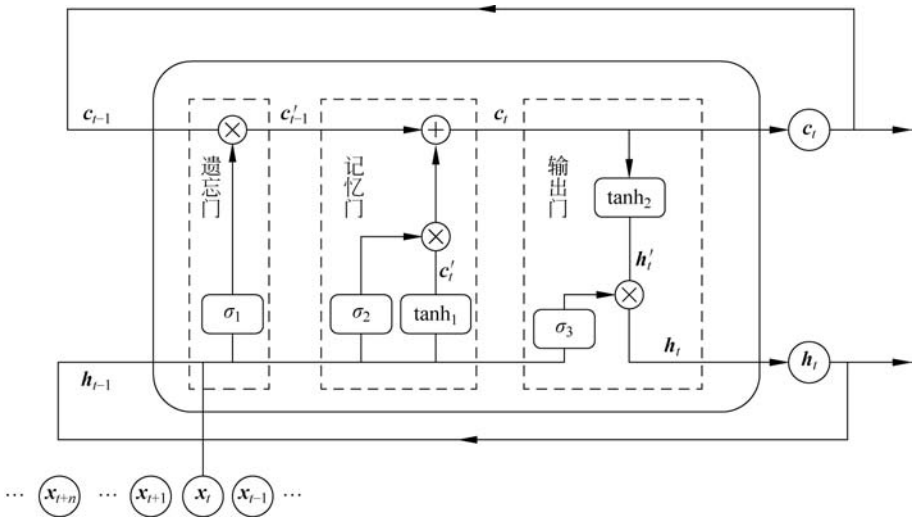


图 5.4 LSTM 流程图

遗忘门、记忆门、输出门就是一系列的激活函数。在遗忘门中，将词向量 $\boldsymbol{x}_t$ 和隐藏状态 $\boldsymbol{h}_{t-1}$（上一次的隐藏状态输出）输入遗忘门的 sigmoid 函数 σ_1 得到值在 0～1 的矩阵并与细胞状态 $\boldsymbol{c}_{t-1}$（上一次的细胞状态）相乘，使细胞状态 $\boldsymbol{c}_{t-1}$ 忘记部分无用的信息得到细胞状态 $\boldsymbol{c}'_{t-1}$；接着，在记忆门中，从记忆门的 sigmoid 函数 σ_2 得到的矩阵与经过 tanh 函数 $\tanh_1$ 得到细胞状态的候选值 $\boldsymbol{c}'_t$ 相乘得到旧的细胞状态需要更新的信息，将该部分信息与经过遗忘门的细胞状态相加得到最终更新的细胞状态 $\boldsymbol{c}_t$；最后，在输出门中，在得到最后的细胞状态 $\boldsymbol{c}_t$ 之后，将其输入 tanh 函数 $\tanh_2$ 得到输出的隐藏状态的候选值 $\boldsymbol{h}'_t$，并与记忆门的 sigmoid 函数 σ_3 得到的矩阵相乘，选择最后输出的部分得到输出的隐藏状态 $\boldsymbol{h}_t$。总之，通过 sigmoid 函数 σ 来选择更新的内容，tanh 函数来创建更新的候选值，来达到更新细胞状态和输出的隐藏状态。

在上述过程中，只讲述了 LSTM 的过程，也就是 bi-LSTM 中的一个方向的过程，而在 bi-LSTM 中，需要把每个词的词向量按照句子的正向顺序和反向顺序分别输入 LSTM，得到每个词的两个隐藏状态并将其拼接就得到了该词的隐藏状态，也就得到词的特征；将这

两个 LSTM 最后输出的隐藏状态结果进行拼接就得到了整个特征。

5.2.3　注意力机制

什么是注意力机制呢？当你在观察一件事物的时候，你的眼睛将会特别关注物体上比较特别的地方，而不自觉地忽略物体其他的地方，这就是注意力机制的由来。这种机制使得你的眼睛可以快速捕捉到关键信息。

那应该如何将这种机制用于计算机呢？在经过神经网络后得到的一系列隐藏状态中，有些隐藏状态是特别重要的，那么它需要特别重视，所以就可以给它比较大的权重，若是不重要的隐藏状态，那么就给它比较小的权重。所以将这些权重写成矩阵形式后（即注意力权重矩阵），再与隐藏状态相乘时，重要隐藏状态对整个和的结果影响更大，不重要的隐藏状态对结果几乎不影响。在神经网络中加入注意力机制的好处是使得该网络可以更好地处理有用和无用的信息以记住更多之前输入的信息。

那这种注意力权重矩阵是如何得到的呢？比如：你知道表示图像子区特征的向量 $\boldsymbol{h}_j$，也就是通过神经网络得到的隐藏状态（图像的特征），也知道了词 i 的词向量 $\boldsymbol{e}_i$，那么直接计算它们的内积 $s_{j,i}=\boldsymbol{h}_j^{\mathrm{T}}\boldsymbol{e}_i$ 并进行归一化 $\beta_{j,i}=\dfrac{\exp(s_{j,i})}{\sum_{k=0}^{K-1}\exp(s_{j,k})}$（$K$ 表示句子中词的数量），这种权重值 $\beta_{j,i}$ 所形成的权重矩阵 $\boldsymbol{\beta}$ 就衡量了每个词与每个图像子区域的相似度，越匹配则它们内积的值就越大。当将子区域图像确定时，假设是子区域 j，那么将与 j 相关的权重值 $\beta_{j,i}$ 与相应的词向量 $\boldsymbol{e}'_i$ 相乘并求和得到的向量 $\boldsymbol{c}_j=\sum_{i=0}^{T-1}\beta_{j,i}\boldsymbol{e}'_i$ 就表示图像 j 所对应的词上下文向量，它表示了每个词对图像 j 的关系密切程度。

5.2.4　网络结构介绍

在 AttnGAN 模型工作原理中，第一步是通过文本编码器中的 bi-LSTM 网络将输入的自然语言句子进行特征提取，得到输入整个句子的全局句子特征向量 $\bar{\boldsymbol{e}}$ 以及每个单词的词特征向量，通过神经网络 F_0（inception_V3 网络）将句子特征转换到图像句子的联合空间，得到隐藏状态 $\boldsymbol{h}_0$，之后图像生成器 G_0 利用隐藏状态 $\boldsymbol{h}_0$ 生成一幅 64×64 低分辨率的图片；第二步将隐藏状态 $\boldsymbol{h}_0$ 和词向量矩阵 $\boldsymbol{e}$（把每个词向量作为矩阵的每一列而形成的矩阵）相乘得到注意力权重矩阵 $\boldsymbol{\beta}$，再与词向量矩阵 $\boldsymbol{e}$ 相乘得到词上下文向量 $\boldsymbol{c}_0,\cdots,\boldsymbol{c}_{N-1}$（$N$ 表示图像子区域的数量），该向量表示了所对应子区域与哪些词有关；接着将词上下文向量 $\boldsymbol{c}_0,\cdots,\boldsymbol{c}_{N-1}$ 和隐藏状态 $\boldsymbol{h}_0$ 一起输入神经网络 F_1（inception_V3 网络），得到隐藏状态 $\boldsymbol{h}_1$；并将隐藏状态 $\boldsymbol{h}_1$ 输入生成器生成 128×128 更高精度的图像；第三步和第二步的过程类似，利用隐藏状态 $\boldsymbol{h}_1$ 和词向量得到新的词上下文向量 $\boldsymbol{c}'_0,\cdots,\boldsymbol{c}'_{N-1}$，将其和隐藏状态 $\boldsymbol{h}_1$ 输入神经网络 F_2（inception_V3 网络）得到隐藏状态 $\boldsymbol{h}_2$，接着将隐藏状态 $\boldsymbol{h}_2$ 输入生成器生成 256×256 高分辨率的图像。这个部分就是整个实验 AttnGAN 由文本生成图像的过程。

在图 5.5 中，用灰色框框出部分是该框架用来训练图中的三个生成器的部分，也就是用来训练生成器的损失函数部分。图 5.5 底部的虚线框是原始的 GAN 利用判别器得到生成器的损失函数部分，在前面的实验中已经有了大致的介绍，兹不赘述。图 5.5 顶部虚线框是

论文提出的另一个损失函数 DAMSM 损失，由于该部分涉及到较多的数学知识，本实验将简要说明该部分损失。该部分是利用文本编码器以及图像编码器得到的词向量以及图像的局部特征作为输入，利用注意力机制，得到图像与词的后验概率以及图像与句子的后验概率，求出这两个分支的交叉熵损失，并将这两条分支的损失与原始 CAN 损失相加，来作最后的损失，通过反向传播来更新生成器的参数，完成生成器的训练过程。

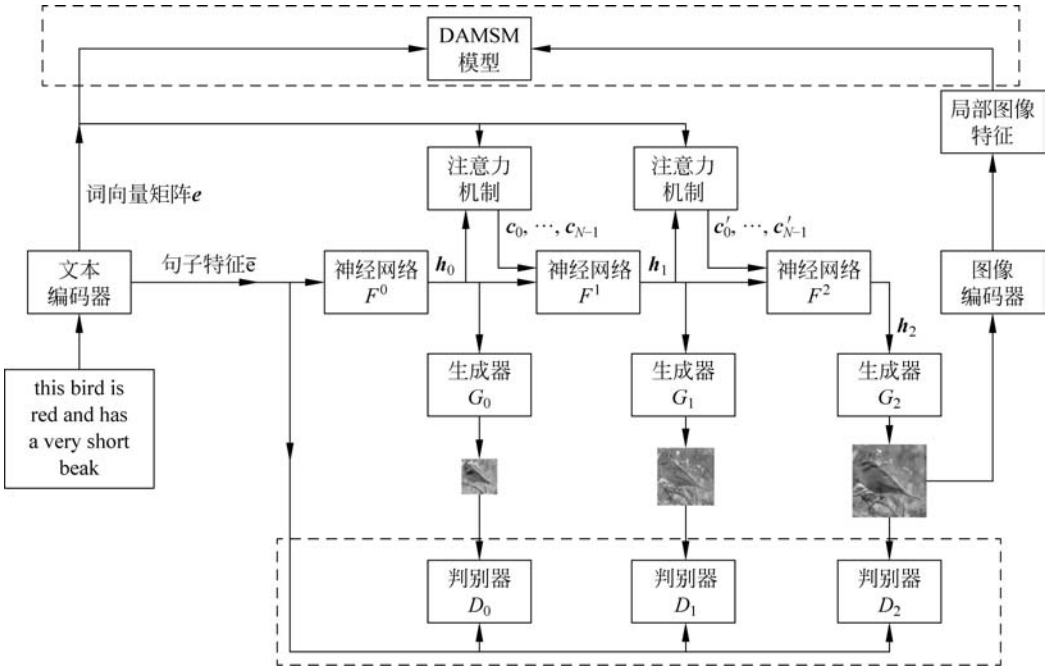


图 5.5　AttnGAN 模型流程图

5.3　实验操作

5.3.1　代码介绍

1. 实验环境

文本转图像实验环境如表 5.1 所示。

表 5.1　实验环境

条　　件	环　　境
操作系统	Ubuntu 16.04
开发语言	Python 2.7
深度学习框架	Pytorch 1.1
相关库	python-dateutil 2.8.0 easydict 1.9 pandas 0.24.2 torchfile 0.1.0 scikit-image 0.14.2

2. 实验代码下载地址

扫描二维码下载实验代码。

3. 代码文件目录结构

代码文件目录结构如下：

```
AttnGAN-master ····· 工程根目录
├── code
│   ├── cfg ····· 训练以及验证模型的配置文件
│   ├── datasets.py ····· 处理数据集中的数据
│   ├── GlobalAttention.py ····· 全局注意力
│   ├── main.py ····· 文本生成图像实验的主程序
│   ├── miscc
│   │   ├── config.py ····· 配置参数文件
│   │   ├── __init__.py ····· 初始化文件
│   │   ├── losses.py ····· 损失函数
│   │   └── utils.py ····· 生成图像的一些主要函数
│   ├── model.py ····· 构建神经网络
│   ├── pretrain_DAMSM.py ····· 用于训练 DAMSM 模型
│   └── trainer.py ····· 文本生成图像
├── DAMSMencoders ····· 存放 DAMSM 训练好的模型文件
├── data ····· 存放 CUB 数据集以及预处理的元数据文件
└── models ····· 存放 AttnGAN 训练好的模型
```

5.3.2　数据集介绍

在本实验中，使用的数据集为 The Caltech-UCSD Birds-200-2011 Dataset（CUB 数据集），数据集展示如图 5.6 所示。该数据集包含了 200 种鸟类的物种，共 11 788 张图像。每张照片都标注有包围框（bounding box）、部件位置（part location）以及属性标签（attribute labels）。

图 5.6　CUB 数据集展示

该数据集一共有 200 种鸟类，种类被编号并保存在 classes.txt 文档中；这 11 788 张图像的名字也被编号并且保存在 images.txt 文档中。每张图片编号和图片所属鸟类物种编

号相对应并且被保存 image_class_labels.txt 文档中。在 bounding_boxes.txt 文档中标注了每张图像中鸟所在的位置，并利用左上角像素坐标以及像素的长宽将鸟框定。

在每张图像中都有 15 个部位用像素位置以及可见性来标注，这 15 个鸟类的部位分别为喙、腹部、喉咙、鸟冠、背部、前额、颈背、眼睛、胸部、整体、头、腿。这些鸟类的部位被编号并且保存在 parts/parts.txt 中。在 parts/part_click_locs.txt 文件中标注了每一张图的每个部位所在的位置以及是否能被看见。

在每个部位上都有不同个数的特征属性，而每个特征属性都有多种不同的表现型。所以在 CUB 数据集中这 15 个部位包含 28 个特征属性，如表 5.2 所示。这 28 个属性总共具有 312 个二值表现型属性。这 312 个二值属性被编号保存在 attribute.txt 文档中。在 attributes/image_attribute_labels.txt 文档中标注这每张图像的每个二值属性是否存在以及存在的可能性程度。

表 5.2 CUB 数据集属性

部位	属性	部位	属性	部位	属性
喙	形状 颜色 长度	背部	颜色 图案	胸部	颜色 图案
腹部	颜色 图案	前额	颜色	整体	大小 外形
喉咙	颜色	颈背	颜色	头	头部图案
鸟冠	颜色	眼睛	颜色	腿	腿部图案
尾巴	上部尾巴颜色 下部尾巴颜色 尾巴图案	翅膀	颜色 图案 形状	身体	下身图案 上身颜色 基础颜色

该数据集的下载可以在项目的网站上，单击图 5.7 所示的 birds 跳转至数据集的页面进行下载。也可以通过 http://www.vision.caltech.edu/visipedia/CUB-200-2011.html 网址进入数据集的网站进行下载。下载的压缩包为 tgz 文件，大小为 1.1GB。

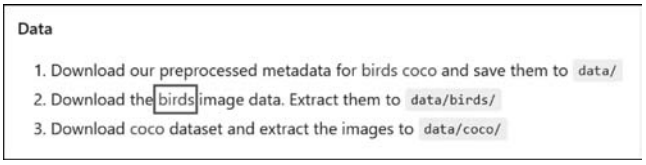


图 5.7 数据集下载地址

5.3.3 实验操作及结果

1. 训练网络模型

（1）下载项目预处理的元数据文件 birds.zip 解压得到 birds 文件（下载地址为 https://drive.google.com/open?id=1O_LtUP9sch09QH3s_EBAgLEctBQ5JBSJ），放入 /AttnGAN-master/data 目录中，在/AttnGAN-master/data/birds 目录下存在一个 text.zip 文件，解压到当前目录即可。

(2) 下载 CUB_200_2011 数据集 CUB_200_2011.tgz 解压得到 CUB_200_2011 文件，将其放入/AttnGAN-master/data/birds 目录中。

(3) 在终端中切换路径到 code 目录下，并开始训练 DAMSM 网络模型：

```
$ cd /AttnGAN-master/code/
$ python pretrain_DAMSM.py --cfg cfg/DAMSM/bird.yml --gpu 0
```

(4) 训练完毕后，得到的模型将保存在一个自动生成的名为 output 的文件夹中，保存在/AttnGAN-master/output/birds_DAMSM_****/中，其中 **** 代指训练开始的时间。例如：/AttnGANmaster/output/birds_DAMSM_2019_11_5_16_12_25/。

(5) 将/AttnGAN-master/output/birds_DAMSM_****/Model/中 image_encoder*.pth 文件以及对应的 text_encoder*.pth 文件(*表示一个数字，例如 image_encoder200.pth 和 text_encoder200.pth)放入/AttnGAN-master/DAMSMencoders/bird/目录下。

(6) 训练 AttnGAN 网络模型：

```
$ python main.py --cfg cfg/bird_attn2.yml --gpu 0
```

(7) 训练完毕后，得到的模型也保存在步骤(4)(5)生成的文件夹 output 中，模型文件在路径/AttnGAN-master/output/birds_attn2_****/中。

(8) 将/AttnGAN-master/output/ birds_attn2_****/Model/中 netG_epoch_*.pth (*表示一个数字，例如 netG_epoch_600.pth)放入/AttnGAN-master/models/目录下并将该文件改名为 bird_AttnGAN2.pth。

到此，训练过程完毕。

2. 使用预训练模型

若是跳过训练阶段，直接利用预训练模型，则执行以下命令：

(1) 下载项目预处理的元数据文件 birds.zip 解压得到 birds 文件(下载地址为 https://drive.google.com/open?id=1O_LtUP9sch09QH3s_EBAgLEctBQ5JBSJ)，放入/AttnGAN-master/data 目录中，在/AttnGAN-master/data/birds 目录下存在一个 text.zip 文件，解压到当前目录即可。

(2) 下载 CUB_200_2011 数据集 CUB_200_2011.tgz 解压得到 CUB_200_2011 文件，将其放入/AttnGAN-master/data/birds 目录中。

(3) 下载 DAMSM 预训练模型文件 bird.zip 并解压得到 bird 文件(下载地址为 https://drive.google.com/open?id=1GNUKjVeyWYBJ8hEU-yrfYQpDOkxEyP3V)，并将其保存在/AttnGAN-master/DAMSMencoders 目录下。

(4) 下载 AttnGAN 预训练模型 bird_AttnGAN2.pth 文件(下载地址为 https://drive.google.com/file/d/1lqNG75suOuR_8gjoEPYNp8VyT_ufPPig/view?usp=drive_open)，将其放入/AttnGAN-master/model/目录下。

到此，预训练的模型已经存放到指定的位置。

3. 测试网络模型

(1) 测试实验的文本输入保存在/AttnGAN-master/data/example_captions.txt 文件中，也可以自己编辑文本并保存其中。但是输入的文本需要符合数据集所说的属性描述，可

以参考/AttnGAN-master/data/attributes.txt 文件，输入自己想要描述的特征以及特征描述句子，否则生成的图像效果会特别差。在原始的 example_captions.txt 中也有作者给出的一些句子样例，读者可以仿照这些样例写入自己想要表达的鸟类特征。

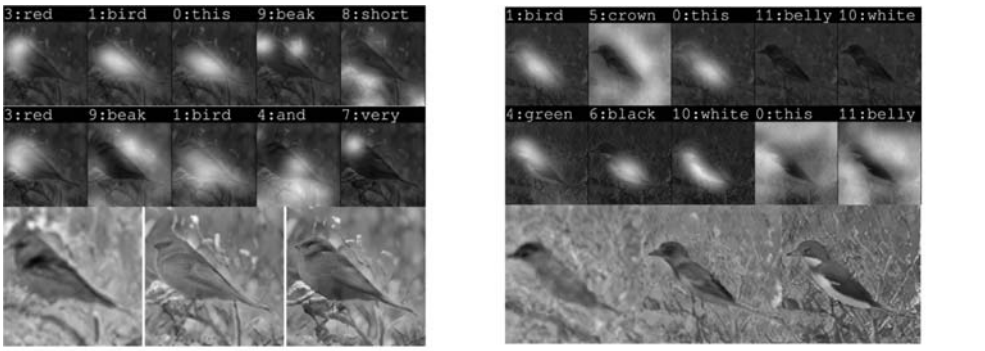
(2) 测试网络模型：

```
$ python main.py -- cfg cfg/eval_bird.yml -- gpu 0
```

(3) 在测试完网络模型后，得到的输出结果保存在/AttnGAN-master/models/bird_AttnGAN2/example_captions/文件夹下。

4. 实验结果展示

在本次实验中，输入的文本为 this bird is red and has a very short beak，得到图 5.8(a)所示的实验结果。输入文本为 this bird has a green crown black primaries and a white belly，得到图 5.8(b)的实验结果。



(a) this bird is red and has a very short beak (b) this bird has a green crown black primaries and a white belly

图 5.8 实验结果图

图 5.8 中的单词表示的是生成器 G_1、G_2 中最关注的 5 个单词排序。图 5.8 底部的三幅图表示的是这 3 个生成器分别生成的分辨率不同的对应句子表述的图像。它们从左向右依次是生成器生成的 64×64 像素的低分辨率图片、生成器在提高分辨率后生成的 128×128 的图像、生成器再次提高分辨率后生成的 256×256 的高分辨率的图像。

5.4 总结与展望

现在主流的文本生成图像的方法基本上都是基于 GAN 的，而纵观至今的文本生成图像的工作，它主要改进方法有三类：改进 GAN 结构；改进文本信息的使用方式；改进用于训练生成器的损失函数模块。从原始的 GAN-INT-CLS 到 StackGAN 再到 StackGAN++，它们通过改进 GAN 的数量，通过堆叠 GAN 的方式生成更加精确的图像；AlignDRAW 加入文本图像对齐的操作，Semantic Layout 将文本和图像提取特征映射到同一个空间，AttnGAN 加入注意力机制，这些方法得到局部图像和词的相似度关系，使得网络生成精度更高效果更好的图像；原始的 GAN-INT-CLS 的损失利用原始的 GAN 损失，之后 StackGAN 开始在损失函数中加入 KL 正则项，TAC-GAN 中加入了分类损失，AttnGAN 中加入了 DAMSM 损失。这些方法都通过加入不同损失的方式训练生成器来得到更加完美的生成器，从而得

到更加理想的图像。

在 AttnGAN 中，利用三层 GAN 加强网络对图像的生成能力；在传统的 GAN 中融入了注意力机制，使得生成器可以更加细粒度地利用输入文本的信息，生成更好效果的图像，同时还加入了一种新的损失——DAMSM 损失，在训练阶段生成器可以兼顾句子尺度的全局图像匹配以及词尺度的图像细节的描述。

而在 AttnGAN 之后，在 2019 年的 CVPR 上，微软研究院发表了新的文本生成图像的论文 *Object-driven Text-to-Image Synthesis via Adversarial Training*[11]。在该论文中，作者提出了目标驱动的注意生成对抗网络(Obj-GANs)，它利用以目标为中心的文本进行复杂场景的图像合成。在两步生成过程的基础上，作者提出了一种新的对象驱动的注意图像生成器，利用文本描述中最相关的词和预先生成的语义布局合成明显的目标。此外，作者还提出了一种基于快速 R-CNN 的目标识别器，以提供丰富的目标识别信号，判断合成的目标是否与文本描述和预生成的布局相匹配。在大规模 COCO 基准上，Obj-GAN 在各种指标上明显优于 AttnGAN 等文本生成图像的最新水平。论文通过对传统的网络注意机制和新的目标驱动注意机制的分析和注意层次的可视化，对传统的网络注意机制和新的对象驱动注意进行了深入的比较，揭示了论文中模型是如何生成高质量的复杂场景的。

5.5　参考文献

[1] Kingma D P, Max Welling M. Auto-Encoding Variational Bayes [EB/OL]. (2013-11-13)[2020-08-20]. https://arxiv.org/abs/1312.6114.

[2] Gregor K, Danihelka I, Graves A, et al. DRAW: A Recurrent Neural Network For Image Generation [J]. Computer ence, 2015: 1462-1471.

[3] Mansimov E, Parisotto E, Ba J L, et al. Generating Images from Captions with Attention[C]// International Conference on Learning Representations, 2016.

[4] Reed S, Akata Z, Yan X, et al. Generative adversarial text to image synthesis[C]// International Conference on Machine Learning, 2016.

[5] Reed S, Akata Z, Mohan S, et al. Learning What and Where to Draw[C]//Conference and Workshop on Neural Information Processing Systems, 2016.

[6] Zhang H, Xu T, Li H. StackGAN: Text to Photo-Realistic Image Synthesis with Stacked Generative Adversarial Networks[C]// International Conference on Computer Vision, 2017.

[7] Zhang H, Xu T, Li H, et al. StackGAN++: Realistic Image Synthesis with Stacked Generative Adversarial Networks [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2018.

[8] Xu T, Zhang P, Huang Q, et al. AttnGAN: Fine-Grained Text to Image Generation with Attentional Generative Adversarial Networks[C]//Computer Vision and Pattern Recognition, 2018.

[9] Mikolov T, Chen K, Corrado G, et al. Efficient Estimation of Word Representations in Vector Space [EB/OL]. (2013-11-13)[2020-08-20]. http://cn.arxiv.org/pdf/1301.3781v3.pdf.

[10] Yao Y, Huang Z. Bi-directional LSTM Recurrent Neural Network for Chinese Word Segmentation. (2016-02-16)[2020-08-20]. https://arxiv.org/abs/1602.04874.

[11] Li W, Zhang P, Zhang L, et al. Object-driven Text-to-Image Synthesis via Adversarial Training Object-driven Text-to-Image Synthesis via Adversarial Training[C]// Internaltional Conference on Computer Vision and Pattern Recogintion, 2019.