老师好，我是汇报人xxx，上周我们介绍了自注意力机制和几种微调模型的方法，最后提到了一种全新的生成对抗网络DragGAN。本周我们简要了解了一下DragGAN和Stable Diffusion模型的区别，然后尝试理解了代码的主要逻辑结构并复现上色任务。最后我们还尝试训练了ControlNet模型。

DragGAN使用了生成对抗网络，它实现了一种新的控制图像内容的方式，允许用户直接拖动图像上的内容来操作图像。用户只需在图像上选定一个控制点和目标点，模型就会将控制点精确的移动到目标点。如果用户不希望图像的其它部位发生变动，用户可以选择使用蒙版，让模型仅绘制特定的区域，也就是下面这张图中的较亮的区域，同时保持图像的其余位置不变。通过这种方式，用户能够精确的控制图像多个维度的属性，包括人物，表情，物体位置，物体形状等。

而Stable Diffusion，是通过文本引导来指定图像生成的内容。但仅仅通过文字，我们很难精确的描述各个物体的具体位置和形状，也就无法对图像进行非常精确的修改。尽管可以通过ControlNet等方式控制Stable Diffusion进行图像生成来尽可能使得图像达到我们希望的样子，但这种控制方式仍然不够灵活，无法将对象移动特定的像素。而DragGAN则允许用户直接将目标拖到目标点来精确的操作图像的各个部分，并且与对象类别无关，也就是说，DragGAN具有很高的通用性，能适用于图像的不同对象类别。它可以成为一个编辑图像的强大工具。下面这些图就是使用DragGAN操作图像的效果。

下面是对ControlNet代码逻辑结构分析。代码的基本流程为，将图像缩放到设定大小，提取图像边缘，处理提示词和权重，进行降噪，然后解码最终生成的结果。

首先是将图像缩放到设定的大小。这里的image\_resolution就是设定的图像分辨率，输入的图像会经过HWC3函数，将图像先处理为标准的RGB三通道图像。也就是说，输入包含透明像素的图像时透明度会在这里消失。然后通过resize\_image函数缩放图像的大小。这里图像的大小不能为任意数值，这是因为在生成过程中，直接处理整张图像的计算量太大，我们需要将图像进行压缩潜空间中，然后在潜空间中进行降噪扩散操作。为了能让这个压缩过程对应上，所以图像大小必须为这个潜空间大小的整数倍。

图像缩放完成后就是提取图像的边缘。这里程序直接调用了cv2库函数来提取图像边缘，这个步骤也就是我们通常说的ControlNet预处理器。不过由于我们给线稿上色的时候，由于线稿本身就可以看作是一张边缘图，因此直接输入线稿的时候其实可以跳过这个步骤。而当我们输入的图像不是线稿的时候，这里就相当于将我们输入的图像先转化成一张线稿再进行上色操作。由于程序处理时通常将黑色视为无，白色是为有，所以边缘图是黑底白线，因此输入通常的白底黑线线稿图的时候需要反色后再输入，或者也可以直接再进行一次提取边缘的操作。

这里img就是待提取边缘的图像，low\_threshold表示，如果像素颜色差异低于这个值，将不被视为边缘。High\_threshold表示如果像素颜色差异高于这个值，将被视为强边缘。这里可以看到这两个参数对提取边缘结果的影响，观察天上这个云附近的结果可以看出，low threshold值越低提取到的边缘越多，high threshold值越高提取到的粗线条就越少。

提取边缘后就是设定随机种子和权重。这里cond变量可以理解为正向提示词，而un\_cond变量可以理解为反向提示词，guess\_mode表示无提示词模式，开启后，ControlNet可以在没有提示词的情况下对线稿进行上色。从这里的代码可以看出，无提示词模式启用后，反向提示词将只走Stable Diffusion分支，不走ControlNet分支，并且此时，ControlNet每层的权重逐渐递增，而关闭无提示词模式时ControlNet每层的权重是相同的。从这些图里面可以看出，ControlNet权重直接影响到最终图像与原图的契合程度，权重低时，最终图像与原图相似度很低，权重高时，最终图像的边缘能很好的契合原图，甚至包括天上云的形状。但同时也可能因为为了过于契合原图，部分物体的颜色和形状会很诡异。

然后就是降噪采样了，这里使用了DDIM采样器。DDIM就是一种降噪模型，根据设定的步骤从一张噪声图中不断的根据设定的条件去掉噪声最终得到图像。然后最后解码生成的图像并返回，完成上色过程。这里ddim steps表示降噪的次数，eta表示随机噪声系数，scale可以通俗的理解为提示词相关性。从这些图里面可以看出，如果降噪的次数设置的过少，图像噪声可能还没去完就输出了导致图像感觉很糊，效果不好。随着降噪次数的增多图像将逐渐变得清晰。Scale表示提示词相关性，这是一个与根据提示词生成的特征相乘的系数，这个值较低时，如果设定的提示词较多，有些提示词的效果可能表示不出来，这时可以调大这个参数。但如果这个参数调的过大，最终图像的颜色会变得很诡异，例如饱和度过高。这有点像训练模型时过拟合的情况。Eta表示随机噪声系数，设置为0时表示没有添加随机噪声，设置为1时表示在生成去噪的过程中会添加随机的噪声来使得最终的结果变得更多样，同时降低可能陷入局部最优的情况。不过这个参数似乎对最终生成结果的影响很小，看下面这些图，除了天上的云彩的形状有细微的差别以外，图像基本没有区别。

然后看到采样的流程，这里conditioning变量表示生成的条件，包含我们输入的提示词和一些设定的参数。Make schedule函数则是用来构造初始化采样器，构造完成后就开始具体的采样操作了。采样前，首先需要生成一张随机的噪声图，这张图就作为初始图像来进行降噪生成。然后将时间步转化为向量后作为条件输入模型。然后就是开始循环降噪的操作了。

这里x表示每次处理的图像，c表示生成的条件，t表示迭代步数，model\_t表示根据提示词和条件生成的特征，model\_uncond表示根据反向提示词生成的特征。这里使用了公式，每次降噪结果等于反向提示词生成的特征加上提示词相关性乘以提示词生成的特征减去反向提示词生成的特征。之前说的scale参数就是在这里起的作用，通过这个公式，图像将朝着设定的条件进行降噪，同时尽量避免反向提示词包含的内容。像这样，不断降噪指定步数后将最终得到结果返回，完成降噪过程。

这里展示的是我们复现代码并完成线稿上色的结果。可以看到ControlNet能很好的完成上色的任务，无论是在有提示词的情况下还是无提示词的情况下。从这张对建筑物上色的结果图中也可以看出，ControlNet很好的理解了图中的各个物品并准确的给其上了正确的颜色。不过似乎还是有少部分物体ControlNet不能正确的识别到，比如这张图天上巨大的月亮，可能线稿图将大小画的过于夸张导致ControlNet没能正确的识别到。

最后我们尝试了训练ControlNet模型，训练用的数据集主要包括三个部分，一个是原图，相当于控制条件，对于线稿上色任务，原图应该为线稿图。第二个为目标图片，也就是上色后的图片。而prompt.json则包含了描述这些图片的提示词。这三部分组成了我们训练用的数据集。由于我们想控制Stable Diffusion用颜色填充这个圆形，Prompt.json里面已经包含了对应的提示词，由于Stable Diffusion模型已经是训练好的，它已经知道了这些提示词的含义，但它不理解原线稿图也就是控制条件的含义，因此我们的目标就是通过训练让它知道这个控制条件的含义然后生成期望的图片。

由于我们缺少合适的数据集，所以直接使用了作者提供的包含五万对图片的fill50k数据集，不过这个数据集并不是仅仅针对训练Canny模型使用的数据集，再加上由于时间的原因，我们的训练时长不够，因此最终的效果并不理想。

我们在batch\_size设置为4，学习率设置为10的-5次方下进行训练的，这里就是我们训练的日志。训练前会复制原来大概7G的Stable Diffusion模型，并输出一个同时包含ControlNet和Stable Diffusion的大概5G大小的模型文件。训练时锁定Stable Diffusion部分，仅训练ControlNet部分，这里就是我们训练时的输出的日志。

这里是使用tensorboard绘制的loss和step之间的关系图像。由于数据集较大，我们训练了一整天的时间，也只完成了3个epoch，总共大概4.5万步。这里表示了每个epoch和每步的loss值，可以看到loss值是在逐渐下降的。

最后是我们使用自己训练的模型尝试进行上色的结果，由于训练时长不够，模型仍然欠拟合，只能勉强看出大致的形状有一定关系。我们可能需要训练更长的时间来取得更好效果。

我的汇报到此结束，谢谢。