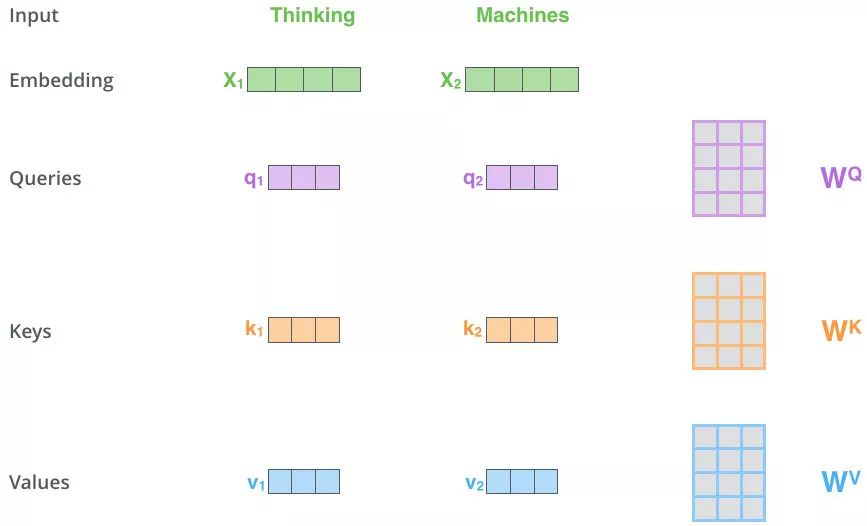
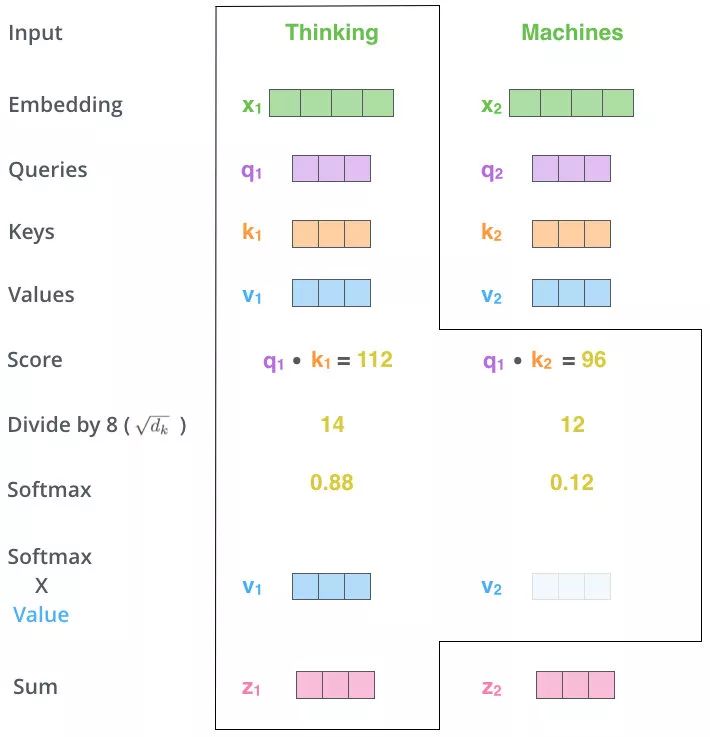
经过了上一周的讨论，我们对于上周提出的问题进行了解决，并且对于论文的“Ralated Work”部分进行了阅读与相关内容的学习，最后总结和提出了对上色任务的见解。

首先是attention机制的具体应用。上周我们已经对attention机制的原理进行了讲解，下面我将先介绍一种特殊的attention机制，self-attention。

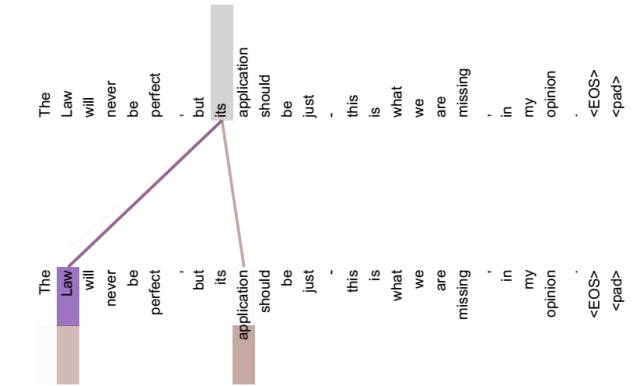


Self-attention的特点在于Q,K,V三者是同源的，都来源于同一个embedding X，如图所示，我们用分别用WQ,WK,WV这三个参数矩阵，对X做线性变换，从而得到Q,K,V这三个向量。



接下来我们将Q和K相乘，得到q1和每个K 之间的相似度，然后需要除以√dk，为了防止

出现后面softmax计算概率分布不稳定的情况，我们可以看到这张图上计算出来的softmax值，一个是0.88，远大于另一个0.12，因为q1和k1都是由同一个embedding转化而来，它们的相似度自然比q1和k2的相似度高，最后我们将softmax乘上v1，得到z1，z1其实就是thinking的新向量表示，它蕴含了thinking machine这句话对于thinking而言其他词的关联程度的信息。

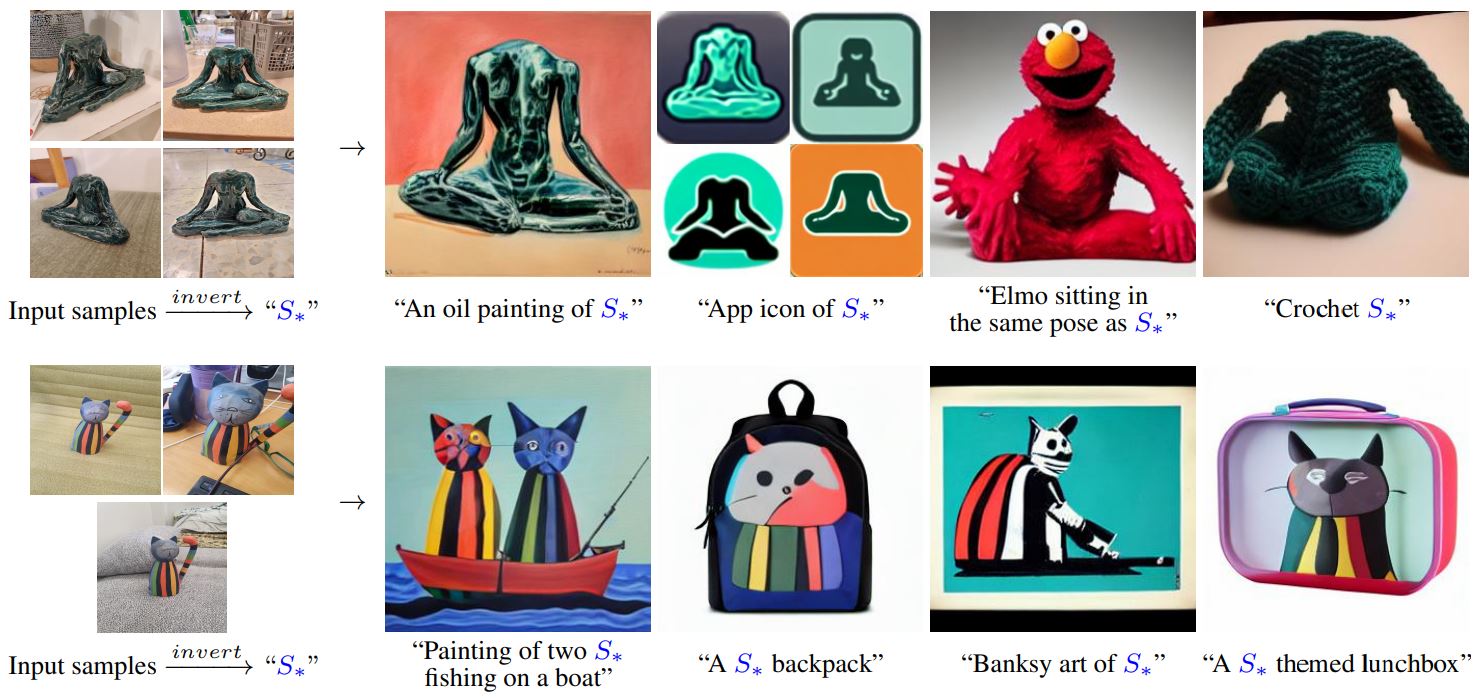


我们看一下self-attention在nlp领域的应用，比如说上面这句话，我们看到右边的its，如果没有注意力机制，计算机可能只是单纯的把its看成一个独立的词，没有任何附加信息，但是通过self-attention后，its的新的词向量会包含一定的law和application的信息，他会与其他词产生关联，从而提高了计算机的语义理解能力。

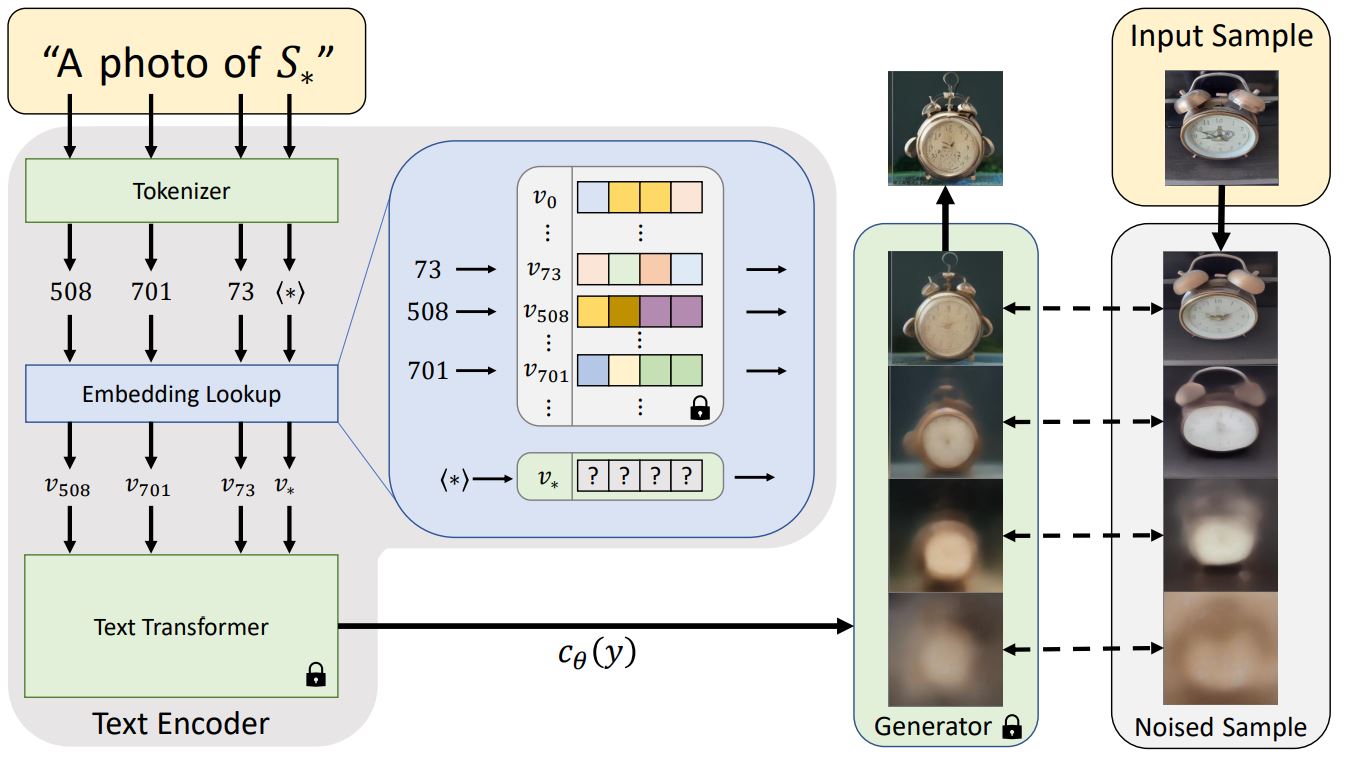
Fine-tuning是指利用已经训练好的模型（通常是预训练模型）作为初始化参数，然后在特定任务的数据集上进行微调，以提高模型在该任务上的性能。通常，预训练模型在大规模数据上进行训练，因此可以提供丰富的特征表示，使得在较小的数据集上进行fine-tuning时可以更快地收敛并提高性能。

目前扩散模型常见的fine-tuning方法有textual inversion, dreambooth, LoRA等等，下面我们来分别介绍上述这三种方法

首先是textual inversion，Textual Inversion是一种从少量示例图像中捕捉新概念的技术。所学习的概念可以用于更好地控制从文本到图像的生成流程。它在文本编码器的embedding space中学习新的“单词”，这些“单词”只是模仿真实单词的一串字母，并不是语言中真实存在的有意义的单词，我们称之为pseudo-words，然后可以在个性化图像生成的text prompts中使用这些pseudo-words，以实现对结果图像的非常精细的控制。

以这张图为例，我们在预训练的文本到图像模型的embedding space中找到描述特定概念的新的pseudo-words，我们用S\*表示。这些pseudo-words可以组成新的句子，将我们的目标放在新的场景中，改变它们的风格或构图，或者将它们植入新的产品中。

我们来看看它的原理



在扩散模型中使用text prompt之前，必须先将其处理为数值表示。这通常是通过将单词转换为tokens来完成的，每个token相当于模型字典中的一个条目。然后将这些条目转换为“embedding”——一个特定的token的连续向量表示。最后，embeddings被转换为单个conditioning code cθ(y)，用于指导生成模型。

通过训练，Textual Inversion为pseudo-word S\*找到了一个最合适的embedding【训练前，如果语言模型原本不认识S\*的话会给他分配一个唯一的新embedding，如果认识的话会有一个固定的唯一embedding】，使用该embedding即可指导原始的扩散模型输出与S\*所匹配的图像。

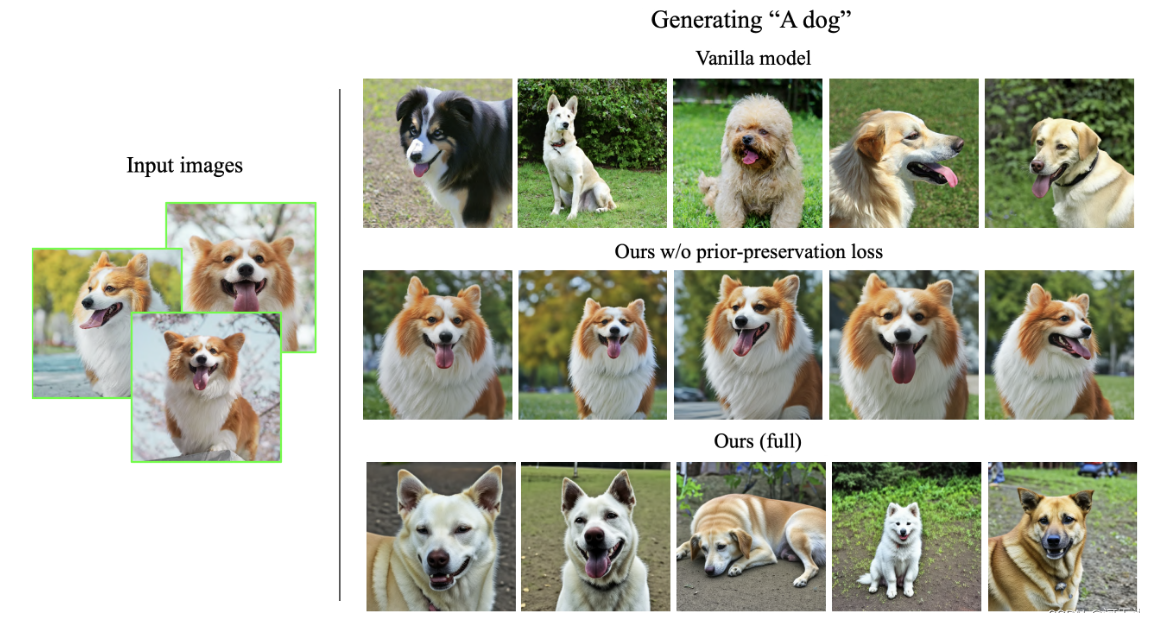
由于Textual Inversion本身并没有修改原始的扩散模型，它并不能教会扩散模型渲染其没有见过的图像内容。Textual Inversion所训练出来的结果实际上就是一个单词与其对应embedding的映射记录，是一个非常小的数据，一般只有十几K。因此textual inversion非常轻量化，但是就总体效果而言还是dreambooth更好一些

下面我们就来介绍下dreambooth

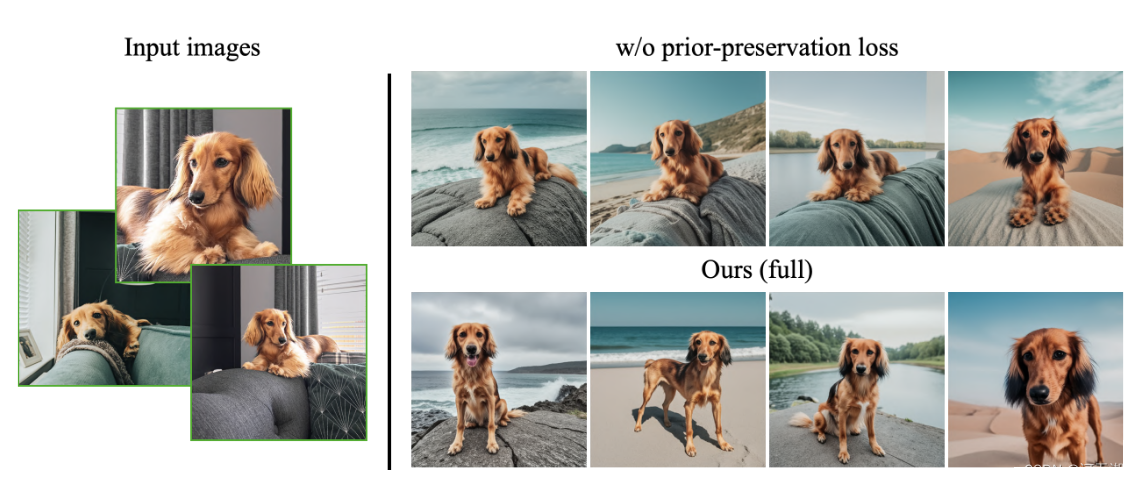


Dreambooth也是一种fine-tuning文生图扩散模型的方法，它最大的特点是subject-driven，针对某一个特定的主体做的生成，这个主体可以是一个动物或者某个物体，包括人都是可以的，我们可以看下ppt上的这张图，我们只需要几张某只小狗的照片，就能生成模型没有见过它做的事情的图片，比如说，去到了希腊，比如说，去游泳，这都是模型没有见到过这只小狗所处的情景，但是模型却能生成出来。这种生成的方式非常的梦幻，所以这个方法就被取名为dreambooth，就是梦幻的照相亭，并且这些生成的图片具有很高的保真度，能够保持这个物体的主要特征。

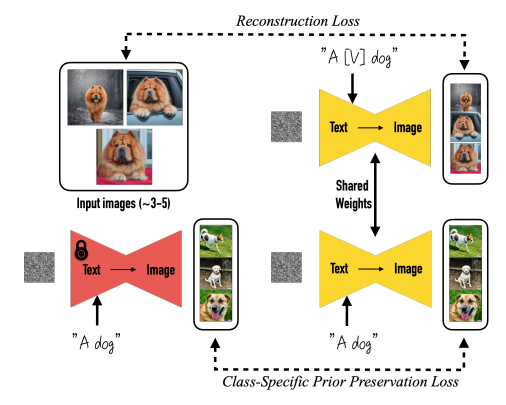
我们来看看具体的实现方法。首先是如何设计prompts，我们把这种文生图的模型比作一种字典，比如说模型建立每个文本和图像的对应关系，这实际上就像一个字典，要使模型认识特定物体，我们需要将该物体添加到字典中。为了让这个物体与字典中的一些词汇产生关联，我们可以使用一个identifier和一个class noun来将该物体与字典中已有的class相关联，这样这个identifier实际上就能和字典里面和这个class有关的其他词产生关联，然后网状的铺开就能让这个identifier能够产生各种各样的词之间的关联，从而使生成的图片多样化。本质上，这就是让模型特定类的prior绑定到这个特定的物体上，模型知道这个特定的物体是哪个类，自然继承了这个类的很多特征，通过这个模型已知的类也能联系到这个类的其他行为，就像我们前面所说的，给定的特定的狗的图片中，并没有它游泳的照片，但是我们通过dog这个大类能够联系到这个文生图模型中狗游泳的样子，自然也就建立起来这只特定的狗和游泳之间的关联，这样就能生成特定的狗去游泳的照片了



Dreambooth和textual inversion不同，它fine-tune模型所有层，而不仅仅是文本编码器，这样就很容易产生language drift的问题，language drift简而言之就是当我们用一个很小的图像样本去fine-tune生成模型的时候，模型对于文本中提到的那些词汇会慢慢的偏移到我们指定的物体上去，而忘记本身的含义，比如说我们输入了这只特定的狗，模型就会慢慢认为狗就是只有这只狗，而忘记它可以代表很多样的狗，通俗讲就是模型学了新的忘了旧的。



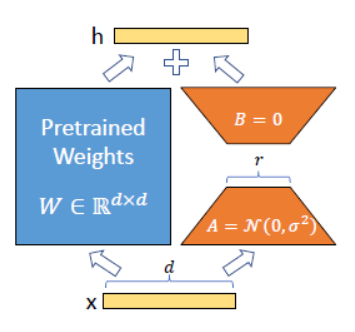
第二种问题是过拟合，我们fine-tune一个模型，很有可能会使输出域局限在特定的物体，特定的几张图片的分布上去，那么就没有办法生成全新的姿势或者结构的一些图片，这种现象经常发生在模型训练时间过长的时候。我们可以看一下这张图片，上面这一排生成的图片都是一种很相似的姿势，所有的狗都是趴着的，这样就没有办法生成多样性的图片。



为了解决以上两个问题，作者提出了Class-specific Prior Preservation Loss，可以让模型继续保持特定类别的prior，本质上来讲，是让模型生成一些样本去监督整个训练过程，以此让模型保持住prior，那么这个类别在模型中的多样性就得到了保证，另外也能够把各种各样的姿势或环境连接到特定的主体上去。具体的方法我们看这张图，训练过程中我们输入了左上角这些图，在普通的训练过程中，他会让这个模型生成图尽量去接近我们的输入图片，这时我们输入的词是A [V] dog，也就是我们之前说的identifier加上一个class noun的形式，这也是扩散模型一般的监督信号，下面这部分就是Class-specific Prior Preservation Loss，在开始训练之前，我们只用这个类别的词去生成一些图片，比如说a dog那就生成了在各种各样环境中的不同狗的图片，然后当训练开始之后，虽然模型的参数更新了，但是我们依然输入a dog，他还是能够生成这些跟训练开始之前的狗一样的狗，而不是去生成特定的狗，也就是说下面左右这两组图片要尽可能接近，越接近就说明模型在训练过程中对dog这个单词的认识越没有发生改变，也就是说模型没有产生language drift。

Dreambooth的特点就是它会训练出一个完整的新模型出来，虽然效果很好，但是会占据相当大的存储空间，我们接下来要讲的lora解决了这一个问题。

Lora，全称是Low-Rank Adaptation，一开始是微软的研究人员为了解决大语言模型微调而开发的一项技术，后来才被广泛应用到视觉领域。Lora采用的fine-tuning方法是冻结预训练模型的权重，然后在旁边加上一些小的网络，最后把预训练模型的输出和新加网络部分的输出加起来得到最后经过调整后的输出，从而达到fine-tuning的目的，同时降低了fine-tuning过程中可训练的参数量，使得训练更高效



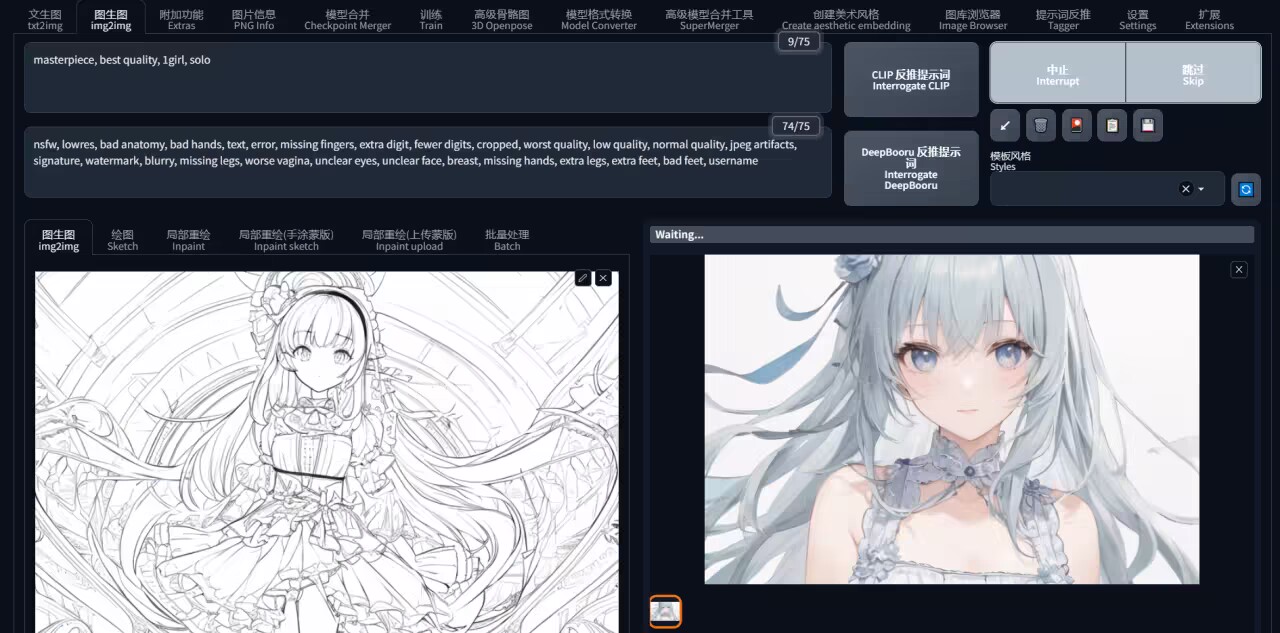
上面这幅图就是lora的一个结构，我们可以看到左边是pretrained weights，这个其实就是预训练的模型，在作者的论文里指的是gpt3，在图像生成领域可以指stable diffusion，然后右边是新加了一个支路，这个支路是去fine-tune这个模型的，这里面的参数是可以改变的，而原来的pretrained weights是不会改变的，右边这个A矩阵是把原来d维度的输入降低到r维度，再经过一个B矩阵升到我们的目标维度，图上所示的就是h的维度d。最后模型的输出就等于原来的输出加上新添加部分的输出，用公式来表示lora的方法是这样的：

其中，W0是预训练的参数，△w表示fine-tuning的参数，两边参数相加起来就是这整个模型，△w又可以被分解成BA，实现的是先降维再升维的操作，在训练开始之前，我们用随机高斯分布初始化A，用0矩阵初始化B，保证训练一开始这个△W依然是0矩阵，这个功能跟controlnet的零卷积层类似，避免在一开始就引入一些噪声，这样就能提高模型的训练速度，使模型更快的收敛。

最后是我们对于上色任务的总结和见解，

上色的过程大致应该是我们输入一张线稿图给controlnet，controlnet通过边缘检测理解图上包含了什么信息，然后在边缘的引导下对线稿进行上色，因为我们的stable diffusion模型一开始是在大数据集上训练的，所以生成出来的图可能比较大众化，有时候我们并不满意这个结果，想要生成特定颜色、特定风格的图片，我们可以将我们的prompts加入进去，通过clip模型来达到自然语言与图像的匹配，并在边缘的约束下将图像创作到其所在的区域，最后达到我们想要的上色效果。

Ai上色与我们一般意义上的人工上色有所区别，人工上色是我们看到一张线稿，理解线稿上的内容并决定线稿每个区域应该填充的颜色，而ai上色是理解线稿上的内容，再由ai根据prompts自由创作和我们的线稿图内容类似的图像，并通过controlnet的canny模型检测线稿的边缘信息来决定生成图像的位置布局，这是一个从无到有的过程。



假设我们不用controlnet，单纯用sd模型，将我们的线稿图输入进去，我们发现模型生成出来的图片只是形状与原图片相似，可以说完完全全是一张新的图片，因为他并没有理解我们图像中具体包含了什么内容，只是根据我们的prompts将原图像魔改成提示词中所要实现的效果。



而当我们加入controlnet之后，controlnet会去检测图像的边缘信息，理解图像每个部分包含的含义，并将此作为一种特殊的prompts传递给sd模型，所以在无参考提示的情况下，sd模型仍能够生成我们所期望的图像，在生成图像的过程中，模型会受到controlnet的引导，使得图像生成在线稿约束的区域之中，总而言之，controlnet在线稿上色中承担了两个作用，一个是提取图像的特征，作为模型的prompts，另一个是控制图像的布局，使其呈现上色的效果。