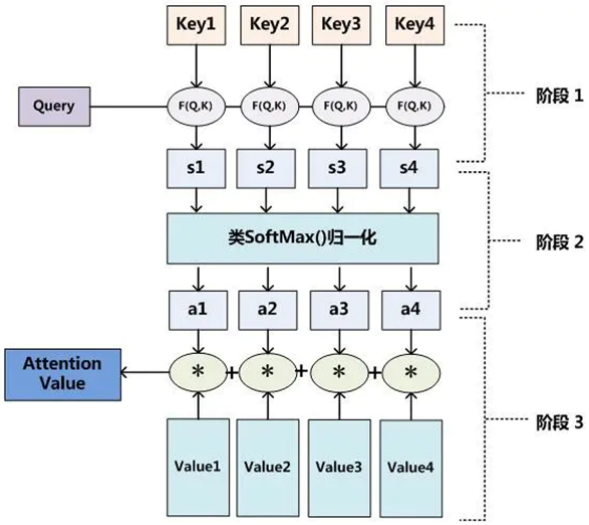
Masked Autoencoders Are Scalable Vision Learners

> 汇报人：张硕　日期：2021.12.2

1. Transformer

1.1 复习Attention & Self-attention



Attention的基本原理：

图书管（source）里有很多书（value），为了方便查找，我们给书做了编号（key）。当我们想要了解漫威（query）的时候，我们就可以看看那些动漫、电影、甚至二战（美国队长）相关的书籍。

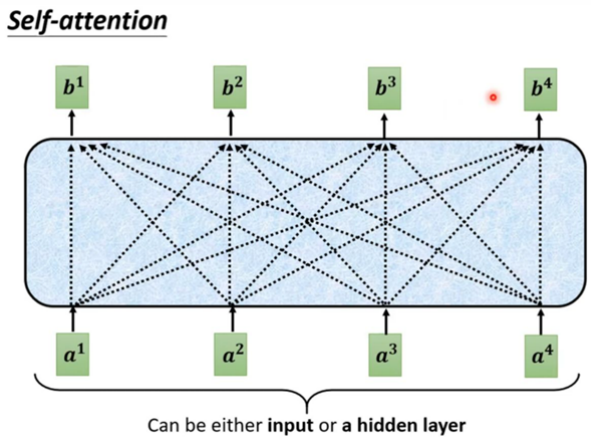
为了提高效率，并不是所有的书都会仔细看，针对漫威来说，动漫，电影相关的会看的仔细一些（权重高），但是二战的就只需要简单扫一下即可（权重低）。

Attention的基础步骤：

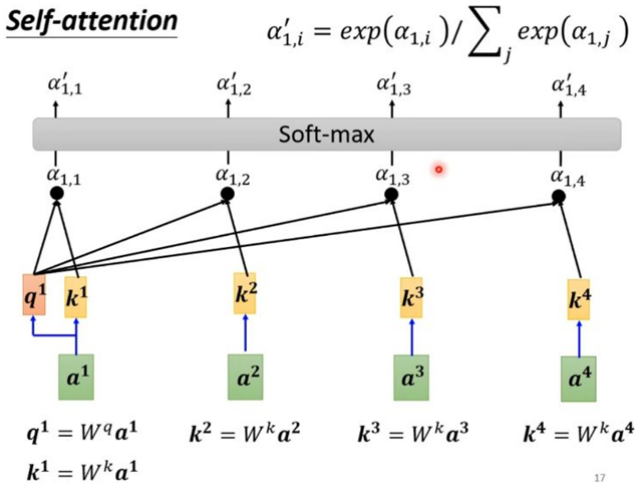
1. query 和 key 进行相似度计算，得到权值
2. 将权值进行归一化，得到直接可用的权重
3. 将权重和 value 进行加权求和

Self-attention:

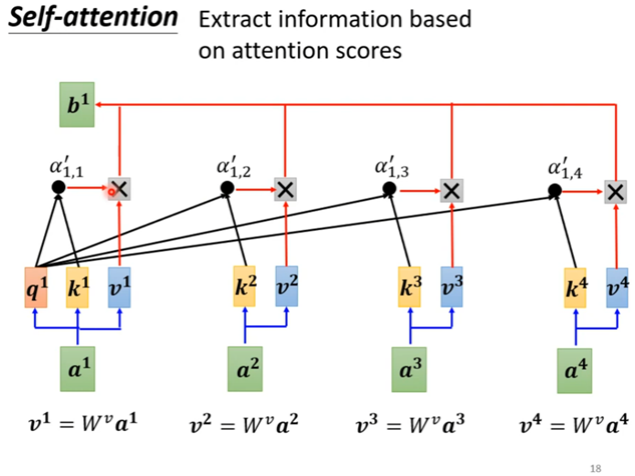
Self-attention的目的可以由下图表示，即通过一个input或者hidden layer计算出输出。因为没有其他外界的输入，所以被称为Self。



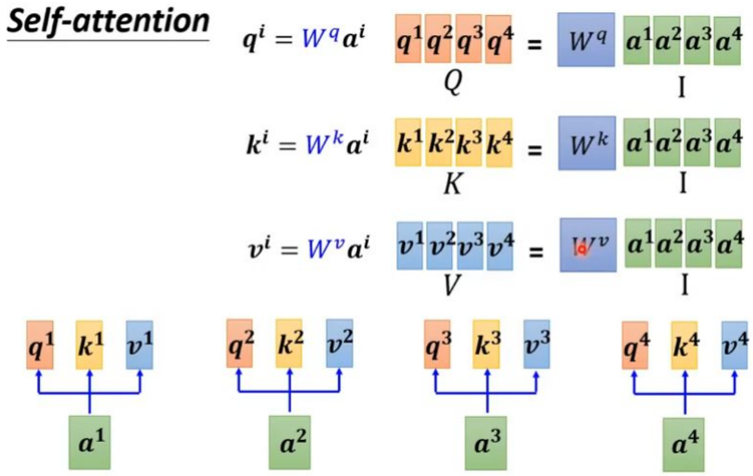
Self-attention的计算步骤与Attention相同，仍然包括以下几步：首先，将query 和 key 进行相似度计算，得到权值，接下来将权值进行归一化，得到直接可用的权重。即下图：



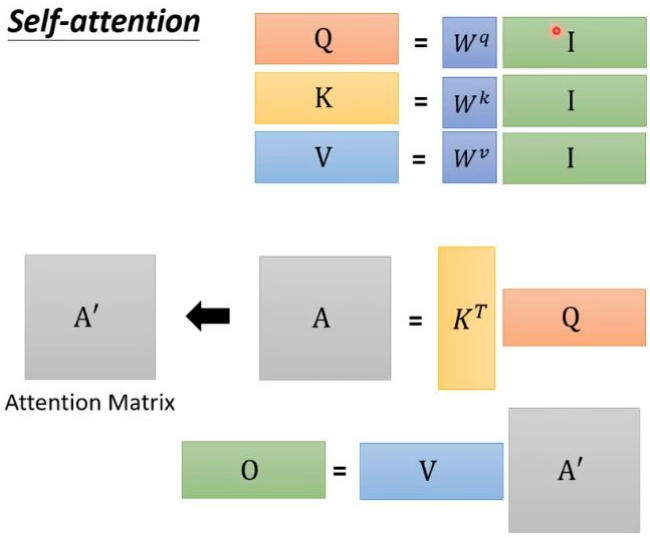
得到相似度之后，要计算得到α后要抽取sequence中重要的信息。根据α就可以知道哪些向量是和a1最有关系，通过这个关系来抽取重要信息，方法是分别计算每个向量的value向量，方法也是乘上W矩阵。然后将这个v值和dot-product得到的α相乘，然后所有向量的该值相加。



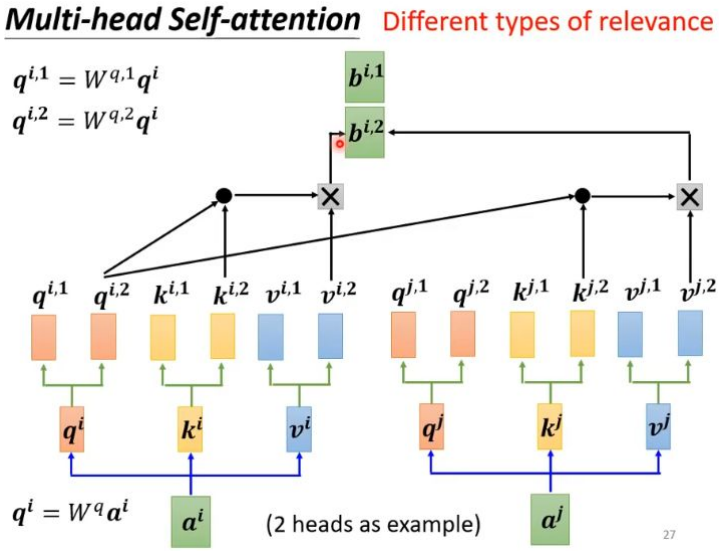
从矩阵乘法角度来讲，计算q向量时，每个输入的vector都乘上矩阵W得到向量q，如果将这些输入的vector都拼成一个大矩阵，那么就是矩阵Q，同理也可以用这种方式得到矩阵K和矩阵V。



将四个输入vector计算α的过程可以通过下图的整合为一个大的矩阵运算过程。**要注意的是k和q相乘后得到的是一个值，所以向量k和q相乘时需要对k做一个转置，所以对于矩阵K也是需要做转置的。**转置后的矩阵K和Q相乘得到矩阵A，就是所有α值拼接得到的矩阵。最后对矩阵做一个softmax即可得到最终的α。最终的输出值b也是用这种矩阵方式得到的。得到的是矩阵O。整理整个过程，输入是矩阵I，输出是矩阵O，整个过程需要学习的**只有矩阵W的参数**，其他过程都不需要学习。

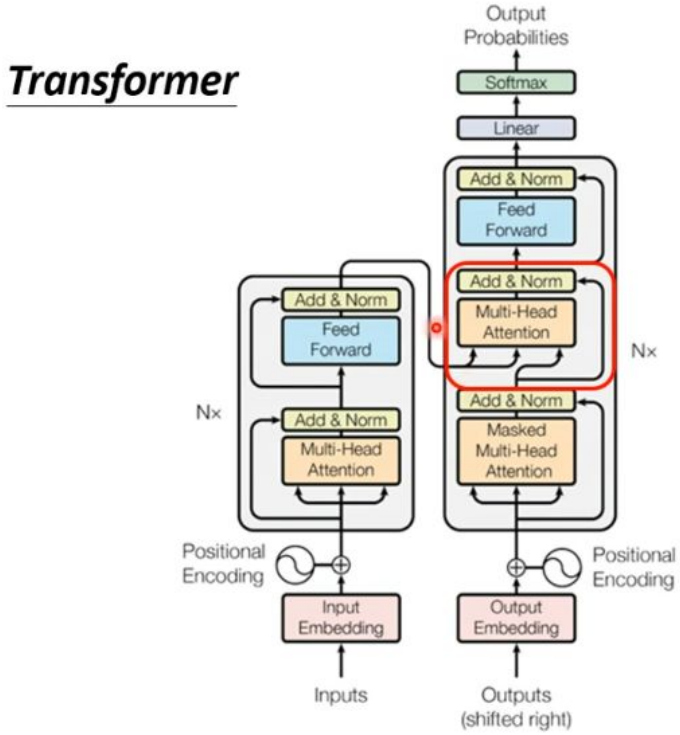


Multi-head:



1.2 Transformer

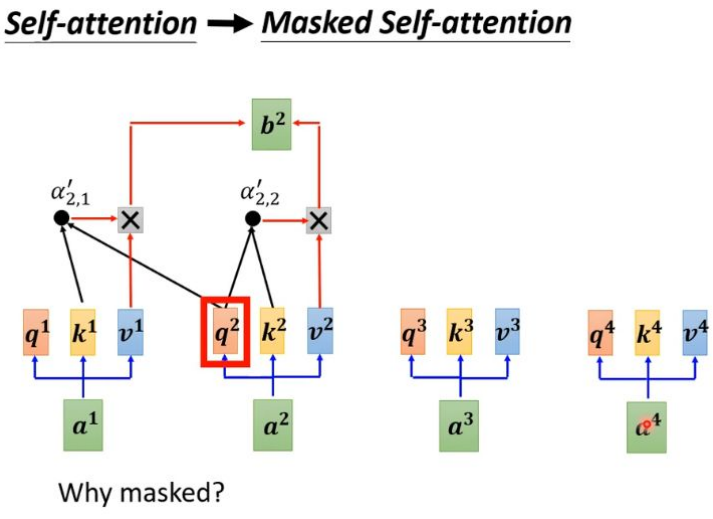
Transformer亦属于seq2seq模型，即输入一个序列，输出不定长序列，例如语音识别中输入信号和输出文字长度不同，翻译任务中输入句子和输出句子也往往长度不同，multi-label任务中一个目标也可以有多个类别。Seq2seq都会有一个编码器和一个解码器。编码器的作用就是输入一排vector，输出另一排vector。一个编码器含有多个block，都是输入输出多个vector，具体如下图：



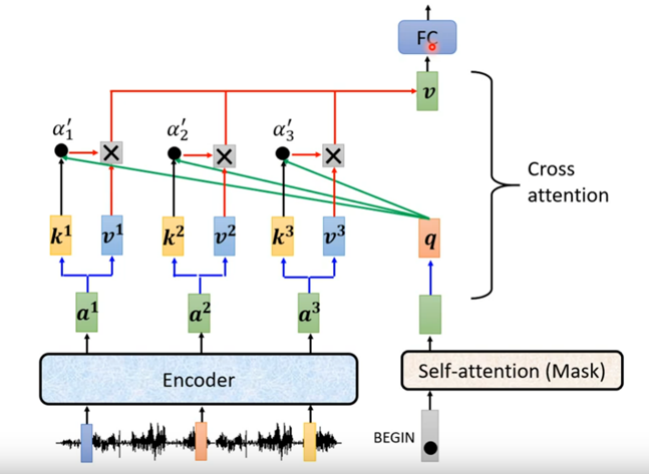
Encoder部分，由N个block组成，每个block在自注意力后和Feed Forward（FC）之后增加了残差层和Norm层。

Decoder部分有两个输入，一个是上一步输出的结果，另一个则是Encoder的结果。Decoder中有两个点与普通的注意力机制不同：

1. 使用了Masked Self-attention，即每次计算输出时，只看已经经过的部分，如下图。也就是说，默认在翻译某一个单词时是不知道下一个单词的。合理。



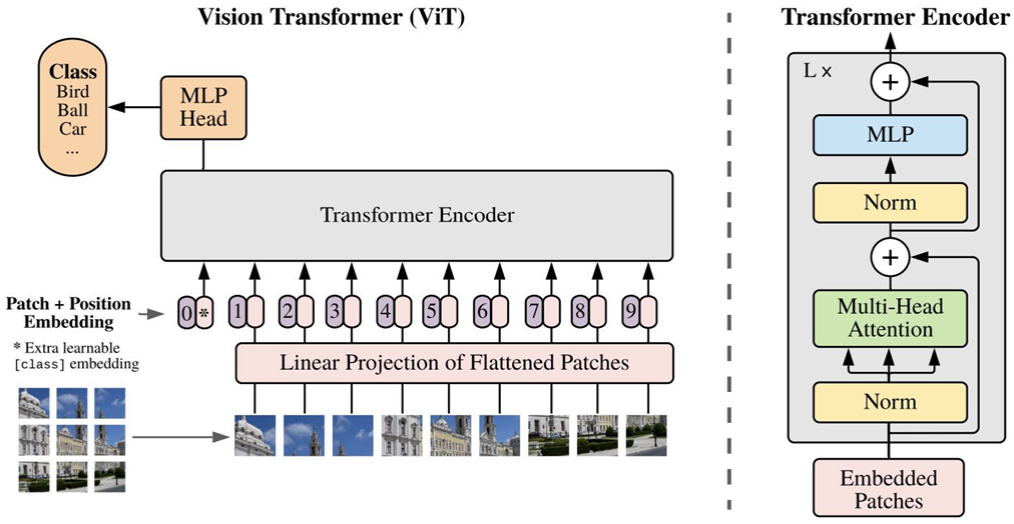
2. Cross Attention: transformer中编码器和解码器的信息传递通过cross-attention来进行，也就是上图transformer结构中的红框部分，在masked multi-head attention处理后的q来自解码器，而k和v来自于编码器。下图就是cross-attention的详细结构。



2. ViT

原文聚焦于迁移Transformer于cv领域。**本文出发点是彻底抛弃CNN，以前的cv领域虽然引入transformer，但是或多或少都用到了CNN或者RNN**，原始直接使用纯transformer的结构并且取得了不错的结果。

在整体的实现上，原文完全使用原始Bert的transformer结构，主要是对图片转换成类似token的处理，将输入图片分成每一个patch，并得到patch Embedding 和 position Embedding，然后就输入标准结构的Transformer去处理。注意，**ViT中的Transformer只有Encoder层，没有Decoder层**，整体的框架如下图所示:



Patch Embedding：对于ViT来说，首先要将原始的2-D图像转换成一系列1-D的patch embeddings，这就好似NLP中的word embedding。输入的2-D图像记为$\mathbf{x} \in \mathbb{R}^{H \times W \times C}$，其中$H$和$W$分别是图像的高和宽，而$C$为通道数对于RGB图像就是3。如果要将图像分成大小为$P \times P$的patches，可以通过reshape操作得到a sequence of patches：$\mathbf{x}_p \in \mathbb{R}^{N \times (P^2 \cdot C)}$，其中，$N = HW/P^2$个patches，这也就是sequence的长度了，注意这里直接将patch拉平为1-D，其特征大小为$P^2 \cdot C$。然后通过一个简单的线性变换将patches映射到D大小的维度，这就是patch embeddings：$\mathbf{x}_\mathbf{p}' \in \mathbb{R}^{N \times D}$，在实现上这等同于对$\mathbf{x}$进行一个$P \times P$且stride为$P$的卷积操作（虽然等同，但是ViT其实是不包含任何卷积操作的）。

Position Embedding：除了patch embeddings，模型还需要另外一个特殊的position embedding。transformer和CNN不同，需要position embedding来编码tokens的位置信息，这主要是因为self-attention是与顺序无关的，每一次都会看全局的信息，即打乱sequence里的tokens的顺序并不会改变结果。transformer原论文中是默认采用固定的positional embedding，但ViT中默认采用学习（训练的）的1-D positional embedding，在输入transformer的encoder之前直接将patch embeddings和positional embedding相加。

到此，ViT的基本思想就讲完了。剩下的东西就是谷歌做了一些实验，验证了这么一件事：**原始的Transformer想在CV任务上达到和Conv同样的性能，必须要使用更多的训练数据。**

3. Masked Autoencoders Are Scalable Vision Learners

> - 论文地址：https://arxiv.org/pdf/2111.06377.pdf
> - 发布时间： 11 November, 2021
> - 作者：K. He, X. Chen, S. Xie, Y. Li, P. Dollár and R. Girshick

3.1 Inspiration

Masked autoencoding of BERT，BERT的预训练范式通过"完形填空"的方式取得了成功，为什么CV不行？作者思考了CV 和 NLP的不同，之后认为有以下三点原因：

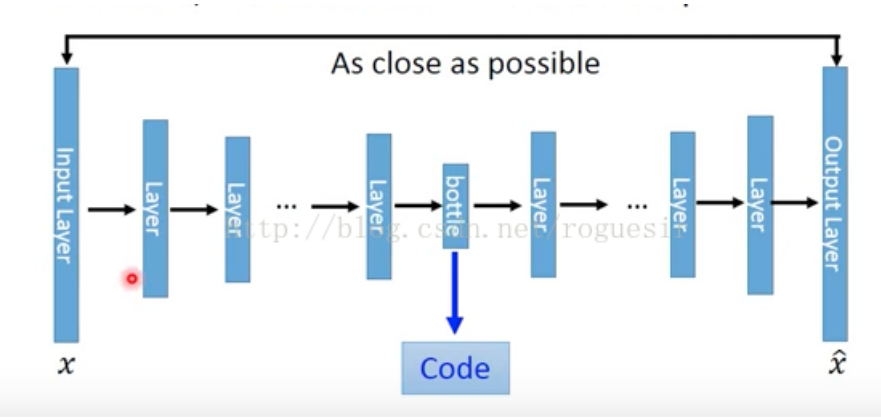
1. **Architecture**：视觉任务中常用卷积网络，将常用的mask token与positional embeddings集成到卷积网路中很难实现。随着ViT的引入，这种情况得到改善。
2. **Information density**：语言具有高度的语义信息特征，而视觉图像具有高度冗余的特点，例如，缺失的patch可以从相邻的patch中恢复。为解决这种情况，提出直接 **mask大部分随机patch**来降低冗余信息。
3. **Decoder**：在语言任务中，解码器的输出预测包含缺失词语的大量语义信息。而视觉中，解码器为了重构像素，因此包含的多是low-level的信息。因此，**解码器的设计**在决定所学习到的隐式特征表示方面所包含的语义信息水平方面起着关键作用。

为了解决上述问题，文中提出了两个关键方法：

1. **A high proportion of the input image**
2. **An asymmetric encoder-decoder architecture**

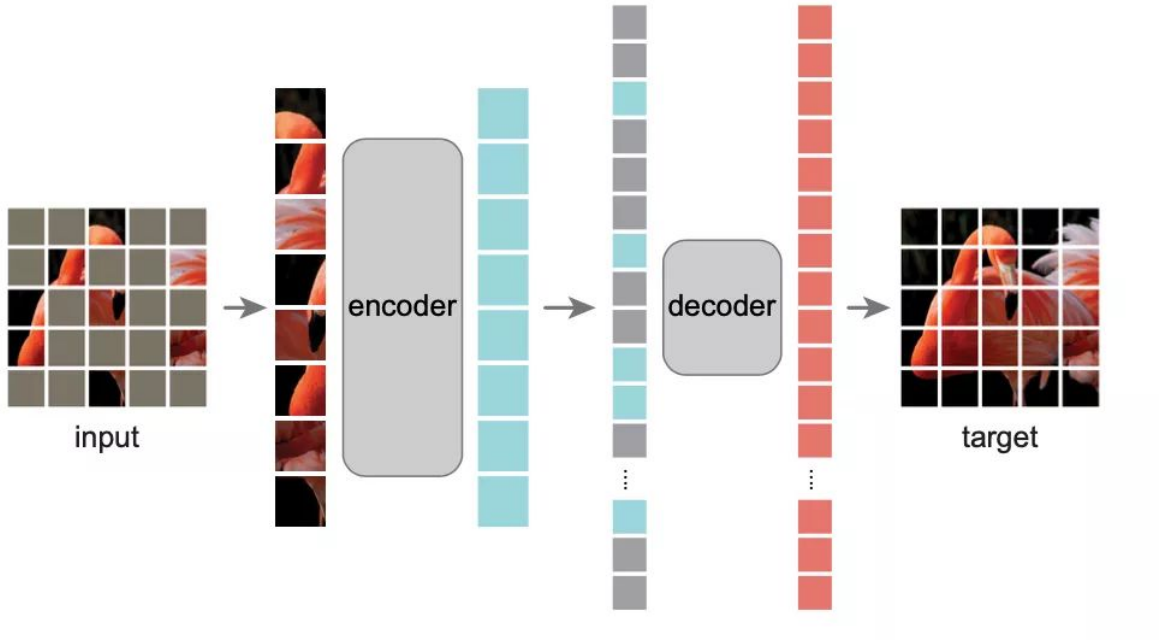
3.2 AutoEncoder

Masked AutoEncoder，其本质是一个AE网络，所以简单介绍一下什么是AutoEncoder。AutoEncoder，即自编码器，以下简称AE。AE是编码结合神经网络构成了一套具有自动编码功能的模型。先编码（图片->向量），再解码（向量->图片），能变回来的那个就是我们想要的向量表示。AE 的目的是是的输入X 和 输出X_head 尽量接近。其结构如下图所示：



3.3 Masked AutoEncoder

MAE与经典的AE不同，它采用了**非对称**的结构设计。在encoder端，它仅对visible patches进行编码，并设计了一个轻量级的解码端，从隐层表示和mask token重建出入图像，整体结构如下图所示：



编码器部分，作者仿照ViT的做法，将图像划分为不重叠的patch，然后对patch进行采样，移除剩余的patch。**采样的方式遵照均匀分布，作者称为random sampling**（作者在后面进行了实验，证明了这是最好的采样方法。）如上文所说，这里采用了很高的采样率——75%，这产生了一个双赢的结果：

- 实验证明，high masking ratio（eg., 75%）结果更好。
- 本文的编码器只需要在整体图像块的一小部分（例如25%）上运行。**这可以节省一定的计算量和显存来训练更大的编码器。**

解码器部分，encoder的输入是所有的tokens: encoded visible patches和mask tokens，它们要加上对应的positional embeddings。解码器的目标是对像素值进行预测，也就是解码器的输出为一系列像素值的集合，输出通道数等于patch中像素值的数量。训练的loss采用简单的MSE：计算预测像素值和原始像素值的MSE（均方误差），不过**loss只计算masked patches。**

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

需要注意的一点是，**解码器仅在预训练期间进行图像重建任务，预训练完成后，仅使用前端的编码器对图像进行特征提取。**因此，解码器可以独立于编码器的设计。在实验中，解码器使用了比编码器更加轻量的网络，进一步减少计算量。