基于卷积神经网络的图像风格迁移

A method of style transfer based on convolutional neural network

学　　院：人工智能学院

专 业 班 级：智能科学与技术 1701 班

学　　号：170407112

学 生 姓 名：薛鑫然

指 导 教 师：于洪霞（讲师）

2021 年 06 月

摘　要

图像风格迁移是广泛应用在图像渲染，个性化定制等领域的技术，传统的数学建模方法已经不能满足风格迁移对多彩化，批量化的要求；而随着卷积神经网络以其特殊的识别过程在图像识别领域被广泛应用，研究者们发现卷积神经网络图像识别方法的自动特征提取可用于模型进行图像风格迁移，相比传统方法，能够极大的提高图像风格迁移的多样化和效率。因此，研究基于神经网络的图像风格迁移对拓宽卷积神经网络的应用，对图像风格迁移的发展都具有重要意义。

本文在介绍了卷积神经网络中卷积层，池化层，全连接层，激活函数以及神经网络训练的基本原理的基础上，给出采用基于 Leon Gatys 创建的风格迁移方法的基本原理，并使用他提出的利用训练好的神经网络提取图像特征的思想，引入 Gram 的概念，分别创建内容和风格呈现的损失函数作为评价标准。使用梯度下降的方法进行损失函数的训练，完成图像风格迁移。

在代码实现阶段，利用 Tensorflow 作为工具，创建了单独的卷积核，池化核函数，搭建了拥有 16 层卷积，3 层池化的神经网络结构，并且创建了使用 Softmax 作为分类标准的全连接层；在数据集的选择上，使用图片数量为 1069 鲜花数据集作为训练数据集，对数据集进行裁剪和按一定比例随机抽取训练验证数据的处理，在训练上，每代选取 16 张数据集，按学习率为 0.0001，在 3000 代后训练最终正确率为 97.5%，得到风格迁移神经网络模型参数；在风格迁移部分，完成了采用训练好的 Vgg19 网络前向传播过程提取图片特征、内容和风格损失函数的构建，使用随机梯度下降的方式进行训练。

在结果中对比了不同风格迁移神经网络参数和风格迁移参数的图片风格迁移结果。不同风格迁移神经网络参数结果中对比了采用鲜花数据集训练的网络参数和 Google 预训练好的 imagenet 参数，结果表明使用不同的网络参数可以产生不同的迁移效果，使用所选择数据集训练的神经网络能更好的呈现内容。在风格迁移参数影响中，对比了损失函数中不同风格与内容的权重的迁移结果，结果表明，不同的权重组合可以实现不同的风格迁移效果。研究者们可以根据实际应用场景的需要和条件，选择适合自己的数据集，调整不同参数实现不同的迁移效果。

关键词：卷积神经网络；图像风格迁移；Gram 矩阵；Tensorflow

Abstract

Image style transfer is a technology widely used in image rendering, personalized customization and other fields.Traditional mathematical modeling methods can no longer meet the requirements of style transfer for colorful and batch; and with the special recognition of convolutional neural networks The process is widely used in the field of image recognition.Researchers have found that the automatic feature extraction of the convolutional neural network image recognition method can be used to transfer the image style of the model.Compared with the traditional method, it can greatly improve the diversification and efficiency of the image style transfer.Therefore, research on image style transfer based on neural network is of great significance to broaden the application of convolutional neural network and to the development of image style transfer.This article introduces the basic principles of convolutional layers, pooling layers, fully connected layers, activation functions, and neural network training in convolutional neural networks, and presents the basic principles of the style transfer method based on Leon Gatys, and using his proposed idea of using a trained neural network to extract image features, introducing the concept of Gram matrix, and creating loss functions for content and style presentation as evaluation criteria.Use the gradient descent method to train the loss function to complete the image style transfer.

In the code implementation stage, using Tensorflow as a tool, A separate convolution kernel and pooling kernel function were created, and based on Vgg 19, a neural network structure with 16 layers of convolution and 3 layers of pooling was built, and Softmax was created as The fully connected layer of the classification standard; in the selection of the data set, the flower data set with the number of pictures of 1069 is used as the training data set, the data set is cutted and the training verification data is randomly selected according to a certain proportion.In the training, each generation select 16 data sets, according to the learning rate of 0.0001, the final accuracy rate of training after 3000 generations is 97.5%, and the style transfer neural network model parameters are obtained; in the style transfer part, the extraction of the forward propagation process using the trained Vgg19 network is completed The construction of the image feature, content and style loss function is trained, using stochastic gradient d

escent.

In the results, the picture style transfer results of different style transfer neural network parameters and style transfer parameters are compared.The results of different style transfer neural network parameters compare the network parameters trained with the flowers data set and the imagenet parameters pre-trained by Google.The results show that using different network parameters can produce different effects.The neural network trained with the selected data set Can better present content.In the influence of style transfer parameters, the transfer results of different styles and content weights in the loss function are compared, and the results show that different weight combinations can achieve different style transfer effects.Researchers can choose a data set that suits them and adjust different parameters to achieve different transfer effects according to the needs and conditions of actual application scenarios.

Keywords：CNN；Style Transfer；Gram matrix；Tensorflow

目 录

第 1 章 绪论

1.1 课题背景

图像风格是指一个图像从一个物体具体的呈现方式到颜色的使用，纹理的形式的多样性的概念，不同的画家在不同的时期对画作的呈现可以有多种风格，后人也不断的在对这些作品的欣赏中将它们划分为了不同的艺术流派。以图 1-1 为例，以下就是两种截然不同的艺术风格。每个人对不同的艺术风格都有不同的见解，所以要如何把一个图像的风格变成另一种风格是一个非常困难难以程式化的工作。对于程序员，特别是机器学习方面的程序员来说，对于风格的模糊的定义是让他们继续研究的阻碍。如何将一个难以精确定义的过程变成一个可泛用的程序，是困扰了很多研究者的问题[1]。

图 1-1 不同的图片风格

随着科技的发展，人们对于图片的操作越来越复杂，图像风格迁移方面的应用也越来越广泛，这是一项利用科技获得名画的风格，然后把提取出来的风格应用在另一张内容图片上的技术。著名的应用 Prisma 通过使用这项技术，可以将普通用户的照片自动变换生成带有艺术家风格的图片。随着深度学习与神经网络技术的诞生与飞速发展，图像风格迁移的应用越来越广泛，例如手机 APP[1]，创作计算机视觉图，时装设计，设计电影镜头，动画风格渲染甚至合成游戏视觉效果等诸多方面。随着越来越广泛的应用，越来越大的规模，使用快速且效果好的风格迁移工具是非常重要的。图 1-2 便是使用像不同风格渲染图片的例子。

图 1-2 不同风格渲染图片

研究者早期寻找图像风格的方法十分类似，这种方法类似数学方法而非计算机方法[1]。先对某一种风格的图像进行分析，建立一个有关此种风格的统计或者数学模型，为了能更符合已经建立的模型，再对需要进行迁移的图像进行加工。这样的缺点非常明显：一个数学模型基本只能对一种特定场景进行分析。所以使用传统方法建模的应用其实有很大的局限性。

这些模型包含提取、绘制等多个独立且复杂的步骤。首先提取图像中物体的边缘信息，接着采用用深度优先遍历的方法将像素点变成笔划路径，逼近绘制出每一条笔划，为了达到不同风格的效果，还需要按照不同的抽象程度来重新绘制线条[1]。

除了上述介绍的建模方法，将 Graph Cuts 理论运用于输入图像前景目标的提取方面是另外一部分研究者所采用的方法。利用交互的分割方式，图像可以被分为背景还有前景。对于背景区域，设计专门的算法，最终能够渲染成艺术图像，获得具有风格化的艺术图像[2]。

总体来说，采用数学建模实现图像风格迁移的步骤太过复杂繁琐，不仅应用的范围不够广泛，效果和效率也不够好。在 2015 年之前，图像风格迁移的研究根本不系统。所以没有受到足够的关注，应用也不够广泛[1]。

随着深度卷积神经网络的出现，这种情况得以改善，这种神经网络学习图像中的高层语义信息后，能够产生强大的计算机视觉系统。在类似物体识别这种采用充足标注的数据训练的卷积神经网络的特定任务中，通过提取图像内容中的高层特征，数据集的泛化能力大大提升，甚至也可以不同类别的图像视觉处理任务完成，例如识别纹理和对艺术风格进行分类[4]。这种利用卷积神经网络实现图像风格迁移实际上是一种转移算法纹理的技术。基于深度学习图像表示的纹理模型，可以巧妙地使用一个神经网络将风格图像与内容图像共同呈现。依靠拥有强

大的内容抽取能力的卷积神经网络（Convolutional Neural Networks，CNN）的发展，图像风格迁移的表现效果得到了明显提升[5]。

进行基于卷积神经网络的图像风格迁移工作有着许多传统的数学建模方法所没有的优势。因为卷积神经网络通过对大量数据集的学习，可以自动的提取特征，而不需要研究者们手动建模，减少了研究者的工作量。同时对大量图像的迁移效果也更好。随着科技的不断发展，图像风格迁移的应用将越来越广泛，所以进行基于神经网络的图像风格迁移的研究是有意义的。

1.2 课题的研究及发展现状

1.2.1 卷积神经网络的发展现状

卷积神经网络（Convolutional Neural Network，CNN）是一种常见的深度学习网络架构。因为科学家们对生物识别认知的深入研究，卷积神经网络便受此影响诞生。20 世纪 90 年代，LeCun et al 等人发表论文，CNN 最初的模型，在后来的研究当中不断的完善它们。他们设计了一种叫做 LeNet-5 的多层人工神经网络，成功地进行了手写数字识别[7]。

LeNet-5 作为较早期的神经网络，在进行手写数字识别时能够直接从原始图片中，识别图片的信息，然而只需通过极少的预处理。然而，由于缺乏训练数据和计算机算力，LeNet-5 处理复杂问题的效果不好。所以人们提出了很多改进，以便克服训练深度 CNN 的困难。其中，Krizhevsky et al.提出了一个著名的 CNN 结构，并在图像识别的准确率上有了很大的提升。他将这个神经网络的结构取名为 AlexNet。它的层次结构和 LeNet-5 类似但更深一些[8]。

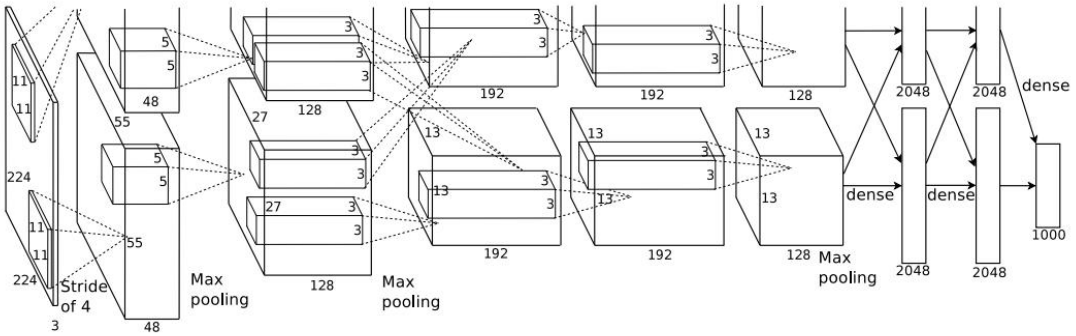

图 1-3 AlexNet 网络结构

AlexNet 的成功带来的是卷积神经网络的研究更多了。在这之后，研究人员提出了许多的改善方法。从结构看，CNN 的层数变得更多了，ResNet（ILSVRC 2015 冠军）的层数是 AlexNet 的 20 多倍，也是 VGGNet 的 8 倍。网络的深度不

断增加的同时，网络利用增加的非线性方法得出目标函数的能力也在大大上升。但是，这样做也使网络的整体复杂程度大大增加，网络难以优化，变得非常容易过拟合。

ResNet（全称残差网络，residual network）是一个 152 层深的 CNN 模型。由微软研究院 KaiMing He 等四位华人在 2015 年提出，斩获了当年 ILSVR 竞赛的图片分类、物体定位、物体检测等多项冠军。先前的深度网络在训练过程中会出现过拟合的问题，会导致丧失一部分信息，而 ResNet 通过在网络中插入跨层连接的机制，把原来的过拟合问题转化为逼近目标函数，这样输入层的信息能直接保留到后面的网络层，一些过拟合的问题就被解决了[9]。

除此之外，每年 ILSVRC 挑战赛以及其他竞赛中也会出现其他许多优秀的模型。例如 VGG、DenseNet、Dual Path Network(DPN)、SENET 等等都在基本的 CNN 模型上进一步作出了不同程度的改进。通过这些模型，神经网络的层数越来越深，但需要的计算量有所减少，取得的效果也越来越好。

随着这么多年的发展，研究者发现 CNN 具有稀疏交互、参数共享的特点，通过多层非线性变换，它自动从数据中提取特征，表达能力和学习能力非常强，对数据的平移旋转和缩放等的适应性也非常好，但是在训练模型时和应用场景等一些方面还是存在很明显的不足之处。例如：需要训练模型的时间非常长，大量计算资源被消耗。即便在 GPU 加速和并行计算的帮助下，训练一个大型深层 CNN 模型也需要数天乃至上月的时间。协调不同机器之间的模型参数的有效同步和更新的挑战巨大。

近日来，美国华盛顿大学和加州大学开发出一种一种基于光学的卷积神经网络加速器，在这种加速器的帮助下，卷积神经网络每秒能够处理拍字节（1 拍字节=2^50 字节）级的大量信息。这项创新利用了光的并行处理性性，预示着卷积神经网络处理大量数据速度不够快的缺陷可能能够得到大大改善，也意味着神经网络能够在更多的新兴领域得到应用，包括无人驾驶这种需要大量并行计算的、5G 计算、数据中心等等。

CNN 已经在诸如计算机视觉等领域取得了巨大的进步，在不断涌现的新技术下，以往算力不足的缺陷也在逐渐改善着。除此之外，还有许多在等着研究人员和开发人员去探索，弥补现有的不足，改善训练的方法，发展已经表现出的优势，弥补现有的缺陷。

1.2.2 基于卷积神经网络的图像风格迁移研究现状

研究者们在见证了卷积神经网络在图像领域的飞速发展之后，也尝试使用该神经网络作为风格迁移的模型，但是在初期还未找到比数学建模更好的方法。

直到 Gatys et al. 在两篇论文中提出了基于神经网络的图像风格迁移方法。通

过 Gramian matrix，Gatys 计算了不同局部特征的相关性，不用手工建模就能生成纹理的方法就此诞生。依靠卷积神经网络（Convolutional Neural Networks，CNN）的发展，借助其强大的内容抽取能力，图像风格迁移的表现效果得到了明显提升[21]。

然而，因为最初的卷积神经网络都是进行数据分类的工作的，它们所做的工作是对图像整体的风格对比，对于细节的处理没有那么精细。所以在几年前，用学习技术实现图像风格化还有很多缺陷[11]。例如，图像风格化后，图像的内容信息中轮廓模糊、色彩的应用不正确、内容细节的还原也存在问题。至于图像的风格信息方面，也有着纹理不够真实、笔触不够清楚的许多不足之处。为了解决上述问题，研究者们需要对图像风格迁移技术进行更多的改进[11]。

在同一时期，在 imagenet 大赛上，Vgg16 网络取得了不错的成绩，研究者们尝试使用这一网络训练。应用 Vgg16 对特征提取内容和风格图像，迭代修改生成图像，最终得到具有指定内容和风格的目标图像。在公开的图像数据集上搭建算法模型，提出 Gram 矩阵的改进方法，通过设计多种艺术风格的迁移实验。最后通过实验验证，这种新提出算法比传统算法实现艺术图像的风格迁移的效果更好，并且所用的时间更短，说明该算法在图像特征提取及图像风格迁移任务上更具优势。

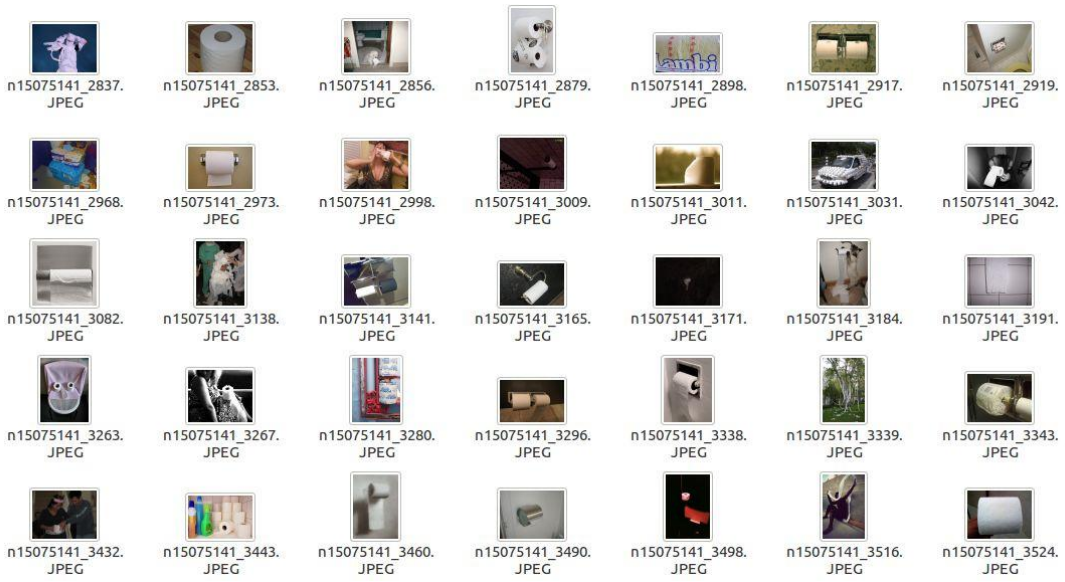

图 1-4 imagenet 数据集

随着神经网络技术更进一步的发展。17 年 ICCV 竞赛 AdaIN 横空出世。AdaIN 的实现过程和之前的任何一种方法都不同，它的主要工作是把一张图片根据特征图，分别提取内容和风格特征，将这两个信息用技术方法分开。AdaIN 的团队认为：从图片中通过神经网络提取的每个通道的特征图中数据的均值和标准差可以

代表这张图片的风格。该神经网络还具有计算内容图像深度信息的模块。通过将迁移图像数据与深度图归一化处理后的数据按元素相乘的方法，突出内容图的深度信息，使得输出的风格迁移图像在深度下具有不同的风格[12]。

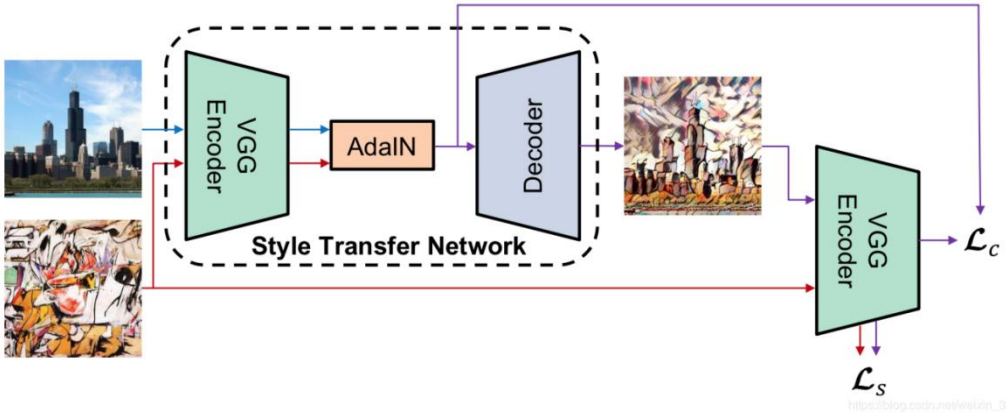

图 1-5 AdaIN 结构

SANet 是 19 年的 CVPR 会议上提出的，它的目的是改进 AdaIN 风格训练迁移充分性。这个结构的提出者们使用深度学习方法中的超维度网络来代替图的风格。通过动态地生成参数，然后将内容图片送入到网络中去完成风格迁移[12]。

随着深度学习的继续发展。近两年研究已不局限于卷积神经网络，通过生成对抗神经网络完成风格迁移的方法也在兴起，Cycle-GAN 网络被研究者用来对图像进行风格迁移，它在无匹配的源图像和风格图像的情况下也被证明是有效的[11]。CycleGAN 网络中的生成器是具有编码器、转换器及解码器等部分，能起到保留原始图像特征和转换图像数据的作用。结果表明，改进后的 CycleGAN 能够比卷积神经网络生成更加逼真的图像，具有更佳的视觉效果[13]。

在最近结束的 CVPR2021 会议当中，来自浙江大学的研究者们又发现了一种完全不需要风格图片的图像风格迁移方法，他们可以在这种情况下，只利用训练好的模型作为指导，便可以将图片迁移至另一种全新的风格，也就是说他们的方法的评价标准不仅仅局限于风格和内容的损失函数，而更多的依赖于源域模型和目标域模型[14]。

图像风格迁移正在朝着更加批量化，更加快速的方向发展着。相信在不久的未来，图像风格迁移能够应用在更多更有意义的领域当中。当今的研究已经证明实现的方式可以不局限于卷积神经网络，随着神经网络技术的发展，相信越来越多的理论和方法能够能够被应用，做更多，更好，更快，更有意义的图像风格应用。

1.3 本文研究内容和章节安排

本文介绍了使用数学建模完成风格迁移存在的一些如步骤复杂，效率低下的种种缺陷，在卷积神经网络出现后迁移效果有了巨大的改善，使用卷积神经网络进行图像风格迁移不仅可一次操作大量图片，训练时间也大大减少，具有很强的研究意义。随后研究了使用神经网络进行迁移的方法。并对比使用自己创建的花朵数据集和官方数据集的参数生成的结果，试图证明在对特定的类别图像操作时，有必要预先训练神经网络。

本文的章节安排：

第 1 章：介绍课题的背景和意义，神经网络的发展。使用卷积神经网络比传统方法的速度、效果更好。

第 2 章：介绍卷积神经网络和风格迁移的基本原理。包括传统神经网络各层的原理，具体的作用，如何训练。介绍如何利用卷积神经网络进行图像风格迁移，如何设置风格和图像的评价标准。

第 3 章：选择数据集，建立神经网络的结构，训练卷积神经网络，并保存网络参数为下一步风格迁移做准备。

第 4 章：使用 Tensorflow，按照原理，构建风格迁移的神经网络模型，设置风格内容的评价标准，使用不同的参数，对比不同的结果。

第 2 章 卷积与风格迁移的基本原理

2.1 卷积神经网络基本原理

2.1.1 卷积神经网络的诞生

早在还没出现卷积神经网络的时候，科学家们对人脑的认知过程已经做了许多研究，也建立了人脑识别图像的基本认知模型。这些研究正是卷积神经网络在将来能够发展并逐步加以应用的关键。

1981 年，来自加拿大的美国神经生物学家 David Hubel 和 TorstenWiesel，通过实验和理论论证，发现了可视皮层是分级的，从而认识了视觉系统是如何处理信息的[14]。

人类的视觉原理如下：从接受原始信号开始，通过初步处理（神经系统发现边缘和方向），抽象（大脑判定，眼前的物体的形状），然后进一步抽象（大脑判断该物体是什么东西）。图 2-1 是人脑进行人脸识别的一个示例，对于不同的物体，人类视觉能认知的最初级的特征是各种笔触和纹理。通过上层的感受系统提取出此类物体的一部分特征，如眼睛。到最上层，在识别系统的帮助下，这些高级特征最后被组合成不同的物体，从而能够让人类准确的区分不同的物体。

研究者通过这个过程，进行了假设：模仿人类大脑的识别物体，构造多层的神经网络，较低层的识别初级的图像特征，若干底层特征组成上一层特征，最终通过多个层级的组合，最终在最上层组合成不同的物体，帮助研究者完成分类。这个假设最终被证明是正确的。研究者们依据这个理论，在传统的神经网络上设计了卷积层，池化层来模拟底层识别初级图像特征的过程。而在顶层，研究者们设计了全连接层来进行分类。综上所述，科学家对人脑的许多研究奠定了卷积神经网络发展的基础，后来卷积神经网络的许多改进与发展也与人脑研究的发展息息相关。

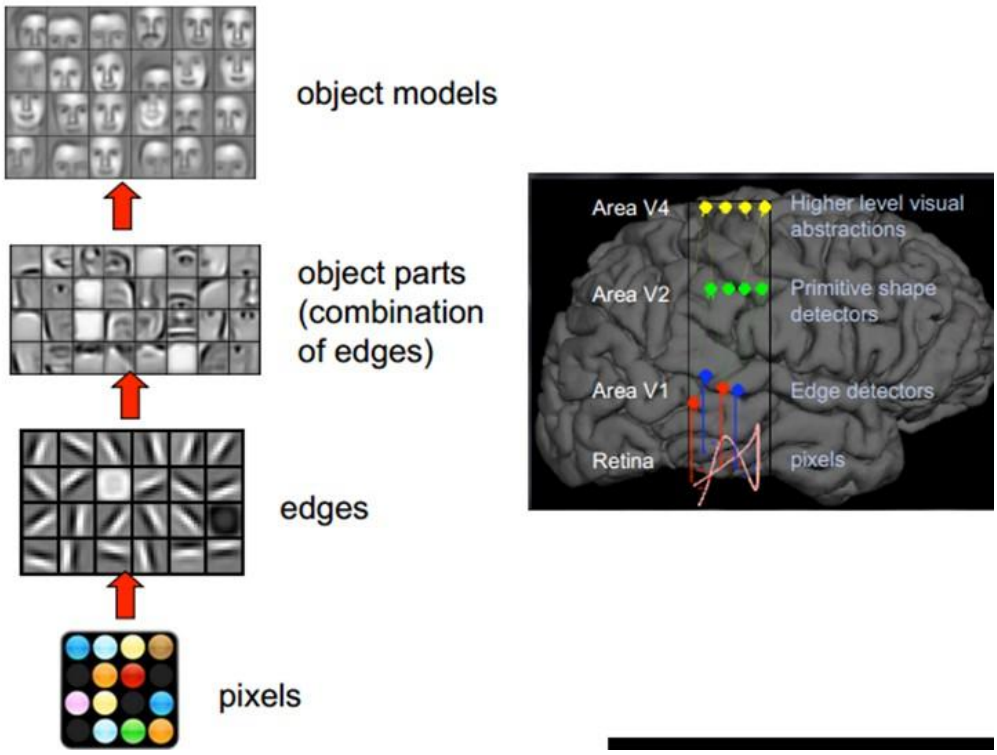

图 2-1 人眼认知的示例

2.1.2 卷积神经网络工作原理

最初的 CNN 的名称是 LeNet，它是由 Yann LeCun 提出并应用在手写字体识别上（MINST）的。结构如图 2-2 所示：

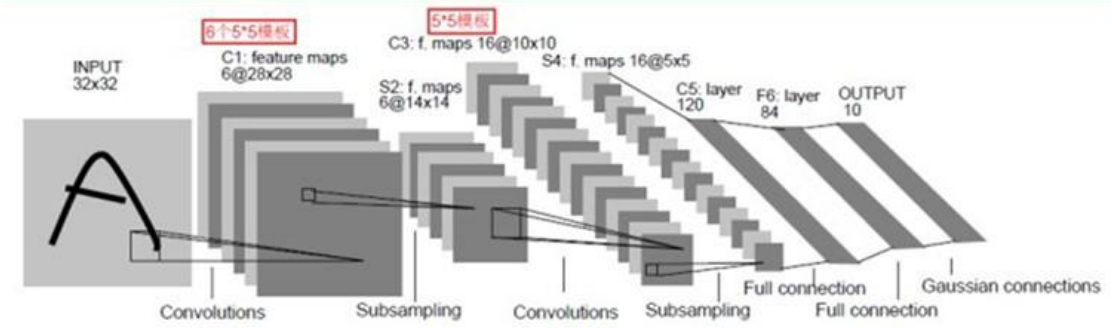

图 2-2 LeNet 结构

LeNet 是一个最典型的卷积网络，由卷积层、池化层，也可以叫做降采样层（subsampling）、全连接层组成。总共有两个卷积层和两个下采样层，卷积层的尺寸是 5*5。它们相互合作用来逐层提取特征，最后通过全连接层的参数完成分

类。卷积层完成的操作，可以认为与生物认知原理中的部分感受概念类似，将大图切割成许多小部分提取特征，而池化层则主要是为了降低数据维度。

综合来说，CNN 通过卷积来模拟大脑识别系统区分特征的过程，降低参数的维度，最后通过传统神经网络中的全连接层完成分类。在 LeNet 的基础上，后续又发展出了 AlexNet，Vgg16，Vgg19。这些网络都是基于 LeNet 的卷积+池化+顶层全连接层架构。

2.1.3 卷积运算

卷积是 CNN 中最基础且关键的步骤，如图 2-3 所示：

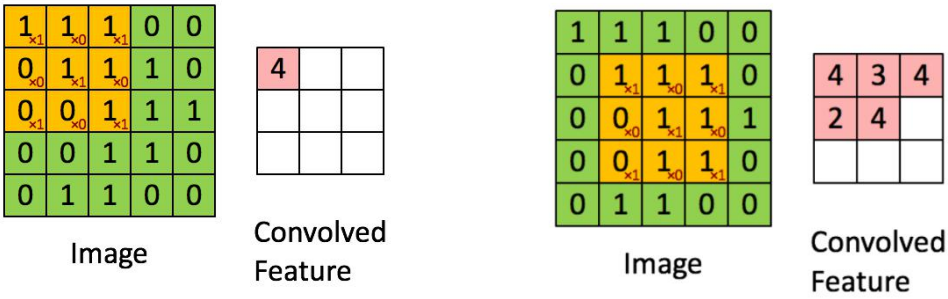

图 2-3 卷积过程示意

CNN 采用的卷积核（kernel）在 x，y 轴上垂直移动，不进行跨越某个单独通道的位移。所以就是每个通道都需要单独的卷积核。如上图所呈现的，绿色表示的为原图像，黄色表示的为卷积核参数，卷积核在图像上移动计算得出一个数作为下一层卷积核对应位置的参数。

在基础 CNN 的每个不同的层中，卷积核的深度都应当一致。卷积需要输入两个参数，这两个参数一个是二维的卷积核，另一个则是输入的二维图片。然后在图像的不同通道中分别计算。以图像风格迁移为例，因为使用的图像是 RGB，也就意味着基本上是三个通道，深度与通道数相等。采用三个独立的卷积核，所以这个卷积核的维度是 $X \times Y \times 3$ 。

假设在某个通道当中输入图像的坐标为(X，Y)，卷积核大小是 $p * q$，卷积核权重为ω，图像的灰度值为 v，卷积过程其实就是卷积核的所有权重在输入图像上对应像素值之和，用公式可以表示为：

$$conv_{x,y} = \sum_{i}^{p*q} \omega_i v_i \tag{2-1}$$

对于不同的神经网络，在卷积方法上和最传统的卷积方法也有不同，以下则是一些不同的完善卷积操作：

（1）采取不同扫描步长（stride），每个扫描的卷积核可以被灵活的规定为不同的步长，但步长的设置不应过大，过大的步长会导致卷积核的感受野损失一部分细节。这种方法固然灵活，但会导致计算的复杂度上升，存储的规模也随之增长。

（2）填充（padding），这是目前被应用的最广泛的完善方法，如果不采用填充的方法，那么卷积过后图像维度是缩减的，在图像周围通过填充 0 或 1 的方法来保证特征图大小与原始图像大小不变。上文介绍的众多神经网络结构均在卷积层中采用了这一方法。

以 stride=1 和 padding 为例，介绍卷积的运算过程。如图 2-4 所示：

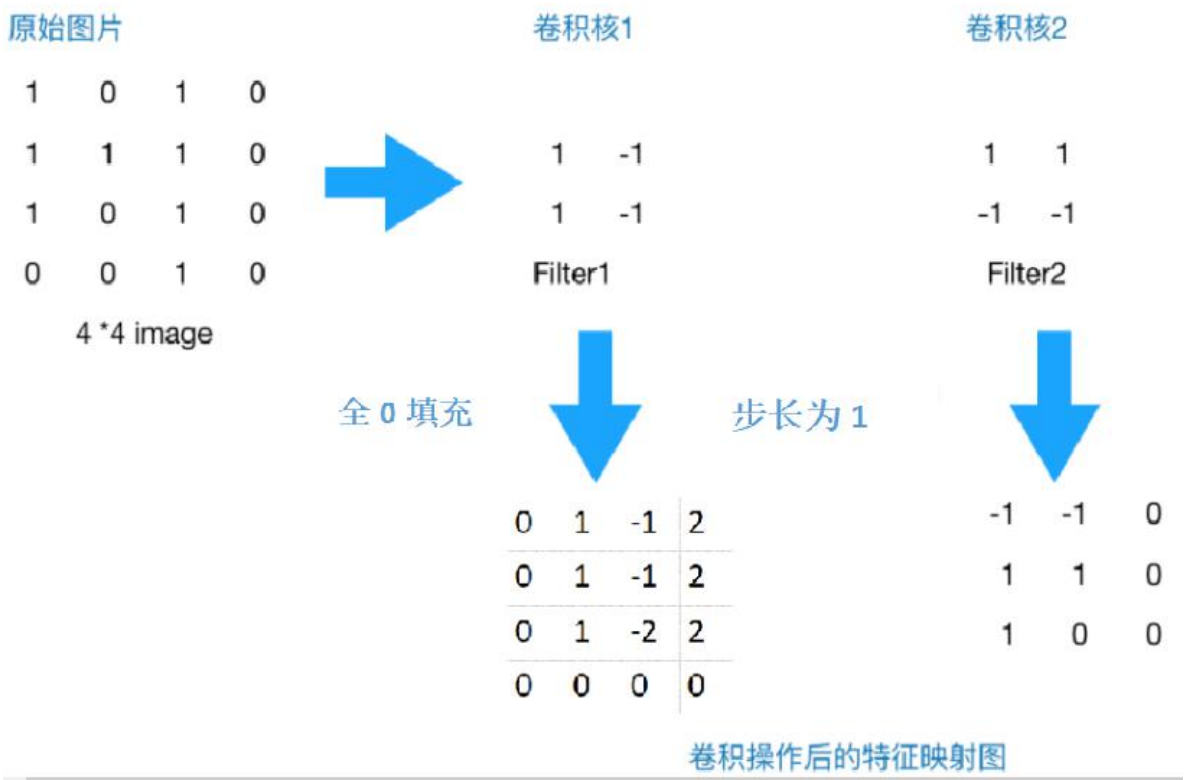

图 2-4 卷积计算示意

对于这个图片大小是 4*4 像素的图像，如果采用两个大小为 2*2 的卷积核来计算。设定步长为 1，即每次以 2*2 的固定窗口往右移动一格。Fliter1 即代表第一个卷积核，其中的参数如图所示，图中的参数与输入图对应的元素相乘。伴随着卷积核的窗口不断的移动，我们可以计算得出一个 3*3 的特征图，而使用全 0 填充的话，则可以得到和原来图片一样大的结果。以此类推，直到计算完这层所有的卷积核。

2.1.4 池化运算

卷积层后是池化层。池化(pooling）是一种降采样操作(subsampling)，主要目的是降低特征图的维度，换种说法，也就是降低卷积之后特征图的分辨率。因为如果特征图参数太多，就意味着拥有过多的图像细节。过多的图像细节对于提取高层特征是很不利的。从数学角度来讲，就是逐渐降低数据空间的尺寸，这样可以减少网络中参数的数量，那么计算资源的费用也进一步减少了，同时也能够有效地控制过拟合。如图 2-5 所示：

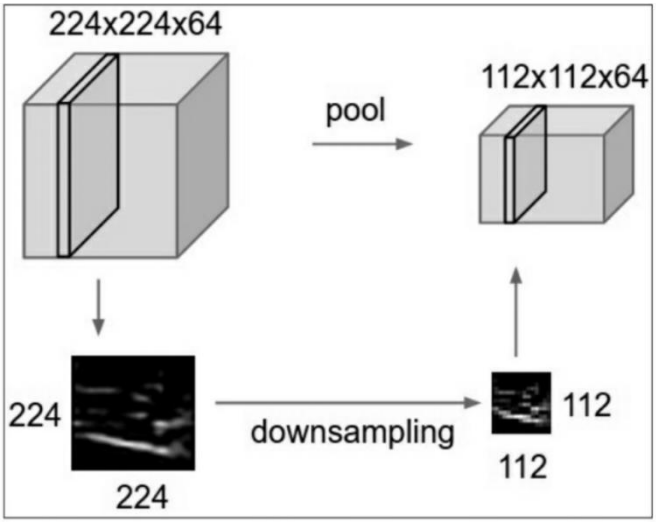

图 2-5 池化操作

如图 2-5 所示：图片经过卷积层后的像素变为 224x224，通道为 64，经过池化层之后，像素变为 112x112，通道数不变，也就意味着图片参数减少为了原先的四分之一，大大减少了计算过程。

目前主要的池化操作有:

（1）最大值池化（Max pooling）：如图 2-5 所示，2 * 2 图片的最大值池化方法就是保留 4 个像素点的最大值。这是目前最常用的池化方式，经过实验与理论的证明，最大值池化被认为是有效的。因为卷积核可以看做是用来提取特征的，不同的卷积核负责提取不同的特征，在实际的神经网络当中有些卷积核提取出 X 方向的特征，而有些卷积核能够提取出 Y 方向的特征，那么如果对其使用最大值池化的方法，提取出的就是两个方向的特征最大值，也就是最能够表征特征的像素值。其余被舍弃的数值对于提取特定的特征并没有特别大的帮助。如果需要后续计算，特征图的尺寸就可以减小，从而减少参数，达到减小计算量，却不损失图像特征的效果。

（2）平均值池化（Average pooling）: 2 * 2 的平均值池化就是取 4 个像素的平均值保留。

继续使用卷积层中提到的例子说明计算过程：

图 2-6 池化运算

通过上一层 2*2 的卷积核操作后，原始图像由 4*4 的尺寸变为了 3*3 的新的图片。池化层的主要目的是通过降采样的方式，压缩图片，减少参数。假设我们规定池化层使用最大值池化的方法，核的大小为 2*2，步长为 1，取计算后每个格子最大的数值进行赋值，那么图片的尺寸就会由最开始的 4*4 变为 2*2。

2.1.5 激活函数

在经过卷积层和池化层的矩阵计算之后，一般会经过激活函数计算之后输出作为下一层的输入。如果不使用激励函数，那么每一层节点的输入都是上层输出的线性函数，意味着输出是输入的线性组合。网络的逼近能力相当有限。然而在使用非线性函数作为激励函数之后，深层神经网络表达能力就更加强大。不再是输入的线性组合。

Sigmoid 和 Relu 是最常使用的激活函数，它们的表达式和图像如下：

Sigmoid：

$$f(z)=\frac{1}{1+e^{-z}} \tag{2-2}$$

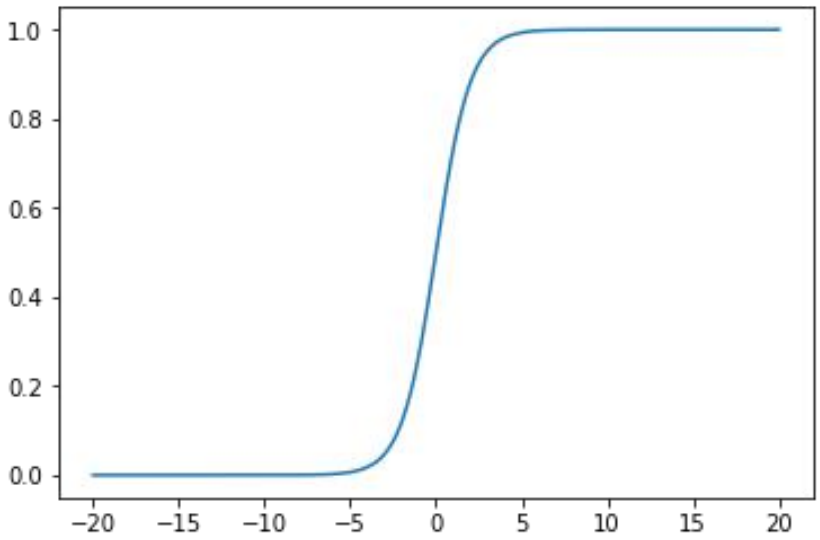

图 2-7 sigmoid 函数

Relu：

$$relu = \max(0, x) \tag{2-3}$$

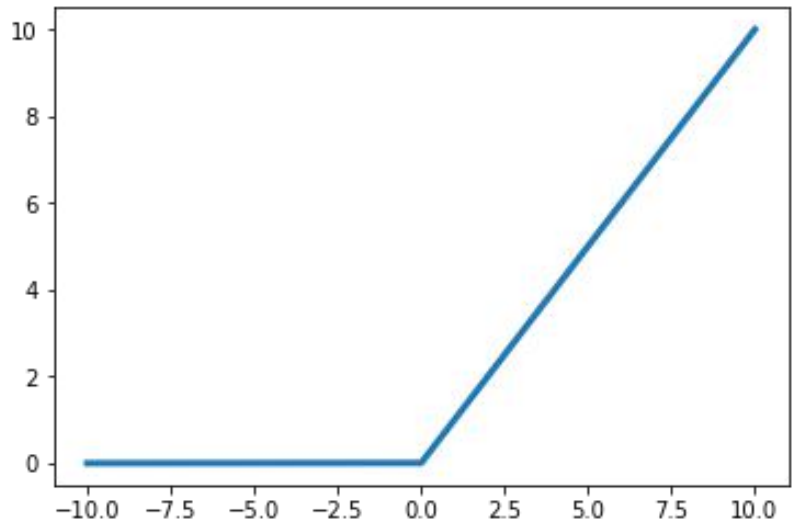

图 2-8 relu 函数

2.1.6 全连接层

卷积神经网络的顶层是全连接层。一般来说，卷积神经网络中的全连接网络(fully connected network)是为了分类。全连接层做的计算是多个矩阵乘法，可以把有用的信息提取整合。全连接层可以把高维变到低维，同时把有用的信息保留下来。全连接网络每一层与上一层完全连接。对于第l层的第 i 个神经元，它的输

出计算公式如下所示。其中w_{ij}代表全连接层中的一个核的参数，$a_j(l-1)$代表上一层卷积核传来的参数，$b_i(l)$代表这个核的偏置。

$$z_i(l) = \sum_{j=1}^{n_{l-1}} w_{ij}(l)a_j(l-1) + b_i(l) \quad (2\text{-}4)$$

经过一个全连接层的参数之后，原本二维的矩阵变成一个数输出。如果考虑激励函数的话：

$$a_i(l) = h(z_i(l)) \quad (2\text{-}5)$$

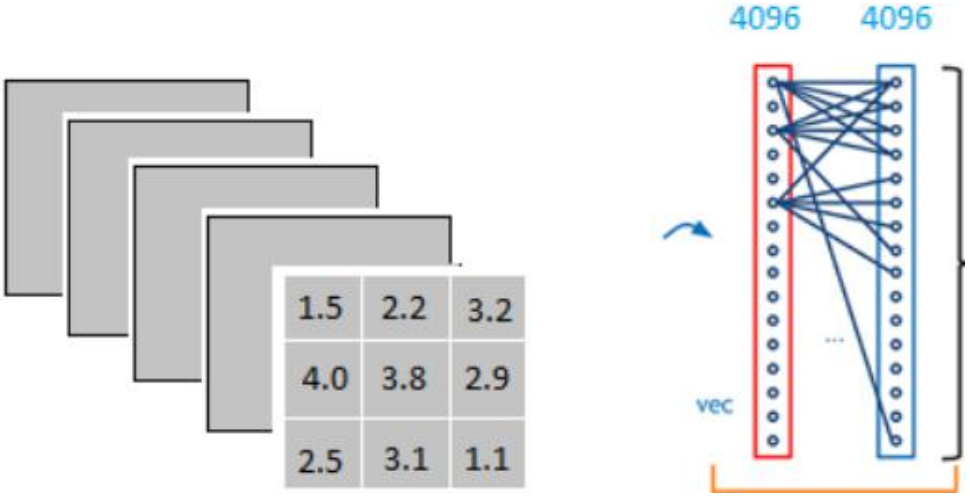

图 2-9 全连接层示意

计算这一层中的所有神经元之后， 作为分类的依据。以图 2-9 为例：神经网络最初的输入为 224x224x3，最后一层卷积可得输出为 3x3x512，如果最后一层是一层含 4096 个神经元的 FC，则可用卷积核为 1x1x512x4096 的参数规模来实现这一全连接运算过程。 每一个卷积核经过全连接层后都会变成一个数输出，经过 4096 个全连接层参数之后，得到 4096 个结果。

全连接层可以看做是一个全局卷积，它把特征的输出整合到一起，输出为一个值。这样可以减少特征位置对分类带来的影响。因为全连接层的存在。使得卷积神经网络能够解决很多非线性问题。分类问题也能很好解决。

卷积神经网络的训练大量的数据集，理论上越多的数据，数据的信息越复杂，得到的效果越好。一般需要先把数据分为训练数据和验证数据，还需要预先设置损失函数和迭代次数。先在训练数据集上训练参数，使用验证验证数据集测试准确率，在一定的迭代次数后，或者损失函数降到一定的阈值之后，完成训练。

2.1.7 神经网络的训练

神经网络的训练需要设置一个目标函数，现在大多的神经网络都使用 Softmax 作为最后一层的评判标准。Softmax 函数可以神经网络最后通过全连接层计算出的结果映射要分类的每一个类别当中，变成此结果对应每个分类的概率。它的公式如下所示，y_i代表最后的概率，e_i代表经过全连接层计算的最终结果：

$$y_i = \frac{e_i}{\sum_j e_j} \tag{2-6}$$

经神经网络预测了类别之后，和标注的类别进行对比，如果一致，神经网络各层的参数不需要改变，如果不一致，则按梯度下降的方式进行更新。

各层参数更新的过程叫做神经网络的反向传播。首先定义误差，它是神经网络输出值和真实值差值的平方：

$$E = \frac{1}{2}\sum_{i=1}^{n}(y_{real} - y_{out})(y_{real} - y_{out})^T \tag{2-7}$$

计算出了误差之后，使用链式法则对每一层的权值进行更新。具体的计算规则如下公式所示，其中：v_j代表每一层输出经过激活函数的激活值，w_{ij}代表在第 j 层需要更新的第 i 个权重，所以这实际上是先对激活值求导，在对激活值求权值的导数。

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial v_j}\frac{\partial v_j}{\partial w_{ij}} \tag{2-8}$$

那么下一时刻的权重更新公式如下所示，η为学习率，不断更新，直到神经网络全部分类正确。

$$w_{next} = w_{now} - \eta\frac{\partial E}{\partial w_{ij}} \tag{2-9}$$

2.2 基于卷积神经网络的图像风格迁移原理

2.2.1 利用神经网络风格迁移的过程

一个训练好的卷积网络在中间的卷积层和池化层里，其实也包含了大量的图像的特征信息，而这些信息，正是风格迁移所需要的。如果将 CNN 的结构，则会发现里面的每一个神经元的其实都呈现了一种不同的特征，靠近底层的，反映的是图像的纹理特征，类似物品的材质。 靠近上层的，则反映是类似物品的种类的实际信息[7]。

基于卷积神经网络的图像风格迁移便是基于此原理，用适当的数学方法，一方面从卷积网络的卷积层和池化层中里提取图像内容有关的信息，一方面提取图像风格有关的信息。

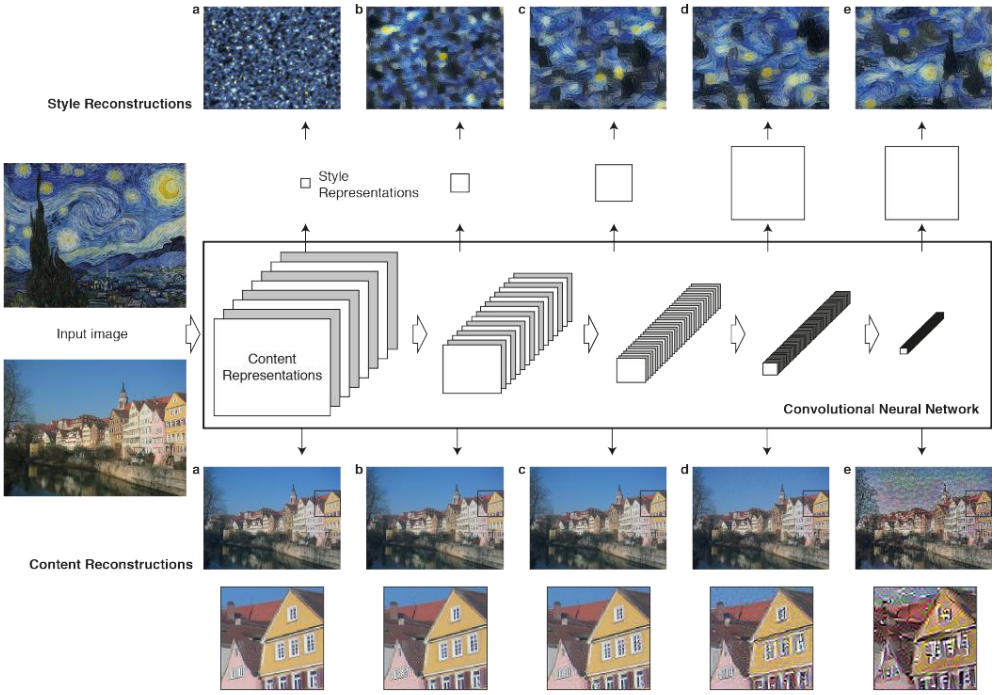

图 2-8 特征提取

以图 2-8 为例。对于风格图片，卷积神经网络的底层提取的是类似颗粒的细节，在神经网络的中层，提取的是类似纹理，笔触的更加大块的特征信息，而到了神经网络的上层，则能够提取整幅图片的风格信息，对于内容图片亦是如此。

进行一张图片的风格迁移的具体过程如下：

（1）用于风格迁移的神经网络的获取。因为图像传递特征的机制是相通的，一个用大量的数据训练好的用来做分类的 CNN，被认为能够对世界大多数图像提取信息。如果有包含内容图片的数据集，可以使用此数据集训练，也可以直接使用其他研究者预训练好的神经网络中的参数。使用不同的参数将会产生不同的结果。

（2）迁移模型的搭建和训练。在图像风格迁移的程序中同样创建和上述同样的网络模型，分别将内容和风格图片进行前向传播，提取特征。创建评价最后生成图片贴近内容和风格的标准。最后设置能够达到风格迁移目标的方法，完成风格迁移。

Leon Gatys 在 2015 年率先提出了卷积神经网络的中间层能够保存图像特征的看法[21]。同时在此基础上，引入 Gram 矩阵的概念，分别创建了评价内容和风

格的损失函数。他使用的是 Google 在大量数据集上预训练好的参数。在训练过程中，他选择了随机梯度下降的方式，实现了任意风格迁移。本课题采用他创建的损失函数，但使用自己创建的数据集训练神经网络。

2.2.2 内容呈现损失函数

内容呈现即生成图像表现原图像中内容的多少。为了评价生成图片与内容图片的差异，设置了内容损失函数作为评价标准。一个好的风格迁移过程应该既能拥有艺术风格，又能还原内容图片中的所有特征。

设在第l层中，特征图的维度为M_l，M_l定义为特征图的高与宽的乘积。卷积特征的通道数为N_l，卷积的高、宽乘积为M_l，那么F_{ij}^l满足$1 \le i \le N_l$，$1 \le i \le M$所以每一层特征图的集合可以表示为$F^l \in R^{N_l \times M_l}$。

我们可以将内容的损失函数定义为$L_{content}(p, x, l)$，F_{ij}^l最初是一张噪声图片x经过前向传播的特征值，P_{ij}^l是原始图像p经过提取的特征值。此公式描述了原始图像P和生成图像F_{ij}^l在内容上的“差异”。内容损失越小，说明它们的内容越接近；内容损失越大，说明它们的内容差距也越大[18]。

$$Y_l = \frac{1}{2}\sum_{ij}(F_{ij} - P_{ij})^2 \tag{2-10}$$

结合所有层，则可以得到总的内容损失函数：

$$L_{content}(x) = \sum_{l=0}^{L} \omega_l Y_l \tag{2-11}$$

2.2.3 风格迁移损失函数

风格的迁移的呈现即生成图像中含有的风格特征的程度。同样的，为评价迁移后的图片呈现了多少风格。创建了风格迁移的目标函数。

引入Gram matrix的概念，Gram矩阵计算的是每个通道i的特征图与每个通道j的特征图的内积，这个值可以看作代表i通道的特征图与通道的特征图的互相关程度[19]。那么利用 Gram matrix 就可以表示每一层各个特征图之间的关系。因此在风格转移的时候，类似人们认知图像的基本方式，Gram矩阵能够帮助我们去寻找基本形状与色彩的组合。

假设我们现在提取梵高的名作星月夜的风格，在神经网络中的特定通道中就有不同的过滤器，也就是不同的Gram矩阵，一些专门提取塔顶这样的尖的能够由我们讲出具体类别的东西，而另一些滤波器专门提取颜色。

接下里要做的工作是将风格的Gram矩阵用数学式表达，Gram矩阵是关于一组向量的内积的对称矩阵，设在第l层中，卷积特征的通道数为N_l，卷积的高、宽乘积为M_l，那么F_{ij}^l满足$1 \le i \le N_l$，$1 \le i \le M$。G实际上是向量组特征的Gram

矩阵。假设某一层输出的卷积特征为10x10x32，即它是一个宽、高均为10，通道数为32的张量。F_1^l表示第一个通道的特征，它是一个100维的向量，F_2^l表示第二个通道的特征，它同样是一个100维的向量，它对应的Gram矩阵如公式2-11所示：

$$G = \begin{bmatrix} (F_1^l)^T(F_1^l)^T & (F_1^l)^T(F_2^l)^T & \cdots & (F_1^l)^T(F_{32}^l)^T \\ (F_2^l)^T(F_1^l)^T & (F_2^l)^T(F_2^l)^T & \cdots & (F_2^l)^T(F_{32}^l)^T \\ (F_3^l)^T(F_1^l)^T & (F_3^l)^T(F_2^l)^T & \cdots & (F_3^l)^T(F_3^l)^T \\ (F_{32}^l)^T(F_1^l)^T & (F_{32}^l)^T(F_1^l)^T & \cdots & (F_{32}^l)^T(F_{32}^l)^T \end{bmatrix} \tag{2-12}$$

利用Gram matrix，我们可以建立每一层的关于风格的损失，其中G_{ij}^l代表待风格转换的特征图的Gram矩阵，A_{ij}^l风格图片的Gram矩阵:

$$E_l = \frac{1}{4N_l^2M_l^2}\sum_{i,j}(G_{ij}^l - A_{ij}^l)^2 \tag{2-13}$$

结合所有层，可以得到关于风格层的总损失：

$$L_{style}(a,x) = \sum_{l=0}^{L}\omega_l E_l \tag{2-14}$$

2.2.4 风格迁移图片更新

在创建了内容损失函数和风格损失函数之后，拥有了评价贴近内容或是风格的标准，下一步需要总体的评价标准将两者结合，并且使用一些方法让损失最小，达到评价标准。

定义如下的函数，$L_{content}$和 L_{style}分别代表内容与风格的总损失，而前面的系数则可以修改，以便让最后合成的结果更加倾向于风格还是内容图片：

$$L_{total}(p,a,\chi) = \alpha L_{content}(p,\chi) + \beta L_{style}(a,\chi) \tag{2-15}$$

采用随机梯度下降的方式让总损失降低。对总损失函数的偏导数实际上是内容损失函数和风格损失函数偏导数的和。根据上面的表达式，对内容层的偏导数计算如下：

$$\frac{\partial L_{content}}{\partial F_{ij}^l} = \begin{cases} (F^l - P^l)_{ij} & \text{if } F_{ij}^l > 0 \\ 0 & \text{if } F_{ij}^l < 0 \end{cases} \tag{2-16}$$

在经过导数计算之后得到的表达式的实际意义其实是内容图与一开始生成的随机噪声图片在对应位置上的偏差，对于每一层，使用现在的特征减去这个偏导数作为内容层的训练。

同样的，按照类似的方法对风格层进行训练。每一层的风格损失对激活函数的偏导数的计算公式为，同样的，保持每一层的参数不变，F_{ij}^l是一开始的随机噪

声图片传递到对应层的特征集合，G_{ij}^{l}代表着该图片在此层的 Gram 矩阵，A_{ij}^{l}则是要转换的风格图片的 Gram 矩阵。两者之和即为总梯度。

$$\frac{\partial E_l}{\partial F_{ij}^{l}} = \frac{1}{N_l^2 M_l^2}((F_{ij}^{l})(G_{ij}^{l} - A_{ij}^{l})) \tag{2-17}$$

我们用每一层的特征减去梯度，再利用卷积神经网络的反传播原理，逐层计算，还原原始的噪声图片，让其同时贴近内容，也贴近风格。正如同随机梯度下降那样，规定训练的迭代次数，或者规定损失的阈值进行停止，即可得到最终相应的图片。待迁移图片总的像素更新公式如下：

$$F_{next}^{l} = F_{now}^{l} - \frac{\partial L_{total}}{\partial F_{ij}^{l}} \tag{2-18}$$

2.3 小结

本章介绍了卷积神经网络的基本原理，包括卷积，池化，全连接层的作用，以及卷积神经网络中激活函数的作用和神经网络的训练过程。

接着利用卷积神经网络的特性，提取网络中的特征，来为图像风格迁移提供工具。同时设置了风格和内容的损失函数作为评价图像风格迁移的评价标准，下一章将通过代码，应用 Tensorflow 作为工具介绍实现过程。

第 3 章 用于风格迁移的神经网络的获取

本章旨在利用 Tensorflow 实现神经网络的训练，完成图像风格迁移的过程。通过使用 Vgg19 网络模型，在创建的鲜花数据上可达到 98%的正确率，拥有良好的提取特征的能力。

3.1 环境介绍

该程序全部使用 python3.7 和 Tensorflow2.1 进行训练。Tensorflow 是谷歌与 2015 年 11 月正式开源的计算框架。Tensorflow 计算框架可以很好地支持深度学习的各种算法，但它的应用也不止于深度学习。

Tensorflow 是谷歌制作的深度学习计算框架。在面世之前它在谷歌内部就已经成功运用在许多方面。利用 Tensorflow 创建的 Inception 模型赢得了 Imagenet2014 年的比赛。Tensorflow 计算速度快，支持的计算平台多，支持的深度算法多而且系统稳定性也很好[12]。

如今，Tensorflow 已经得到了更广泛的应用。如今，谷歌的语音搜索，广告，图片，街景图，翻译等众多产品都是基于 Tensorflow 制作的。谷歌的 DeepMind 也使用 Tensorflow 作为今后继续研究的工具，这个团队后来制作出了著名的 AlphaGo。

除了在谷歌内部大规模使用，Tensorflow 也受到了企业人士和研究人员的广泛好评，现如今，已经有 1500 多个 github 的代码库使用了了 Tensorflow，其中谷歌官方提供的只有 5 个，说明人们在不断地改进其功能。包括优步（Uber），Twitter，京东，小米等国内外科技公司也纷纷加入使用 Tensorflow 的行列[12]。正如谷歌在 Tensorflow 开源原因中提到的那样，他们制作的是一个工具，使得学术界可以更方便地交流学术研究成果，工业界可以更快地将机器学习应用于生产之中。图 3-1 为 Tensorflow 的部分应用。

图 3-1 Tensorflow 的部分应用

本程序基于 Tensorflow 基于 2.0 版本，只需进行少量代码就能获得出色的性

能，同时提高了在 GPU 上的性能表现。让我们可以在原来的基础上，使用 Tensorflow 2.0 可以快速的完成迁移，进行模型的训练任务。还有一个可以值得一提的点是，如果我们使用 GPU 进行训练，表现会比 CPU 好很多，甚至有几倍的速度提升。

3.2 Vgg19 网络结构

在下文的基于卷积神经网络的风格迁移过程中，我们将使用 Vgg19 网络作为风格迁移的结构。VGG 是 Oxford 的 Visual Geometry Group 提出的。该网络是 ILSVRC 2014 上首次提出的[24]，在当年的数据上获得了正确率 98%的成绩。同时也证明了神经网络的深度在一定程度上与最终性能有关。

使用 Vgg19 作为迁移的网络有如下的优点。其一是因为 Vgg19 的卷积核和池化层共享参数，这就使得该神经网络不仅在特征提取方面拥有优秀的性能，修改参数也变得非常方便。其二是因为 Vgg19 没有非常深的深度，保证了训练不需要特别强大的计算机性能。

Vgg19 包含了 19 个隐藏层（16 个卷积层和 3 个全连接层），如图 2-10 所示。前两层使用了两个深度分别为 64 和 128 的卷积核和池化层作为一组，后三层变为四个卷积核和一个池化层为一组，深度分别为 256，512，512。结构非常简洁。整个网络都使用了同样大小的卷积核尺寸（3x3）和最大池化尺寸（2x2）。因为图像风格迁移不涉及到图像的分类，而 Vgg19 的全连接层是用来进行这项工作的，所以我们在具体的实现过程中会将全连接层去除。

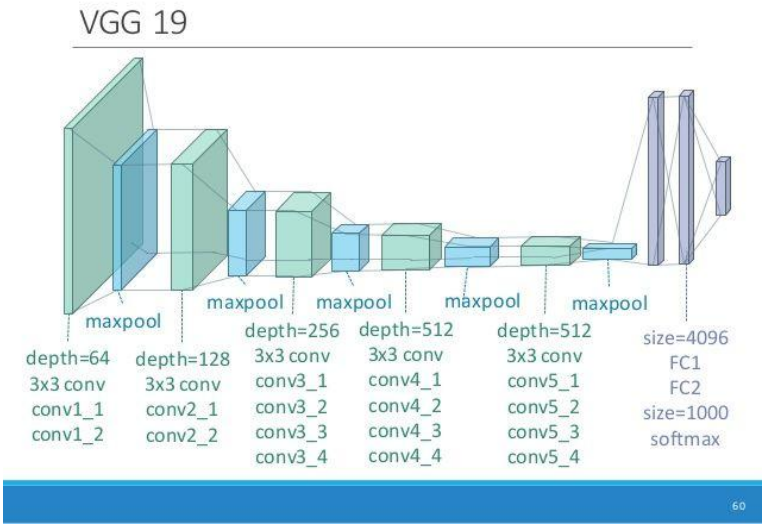

图 3-2 Vgg19 结构

在 VGG 中，使用了 3 个 3x3 卷积核来代替 7x7 卷积核，使用了 2 个 3x3 卷积核来代替 5*5 卷积核，这样做在保证了神经网络在具有相同感知野的情况下，网络的深度有所提升，从而性能也有了改善。虽然 vgg 使用了更多的参数，从而耗费了比较大的计算资源，可能会占用更多的内存，但是总体来说优点大于缺点。

3.3 神经网络结构的搭建

3.3.1 卷积核的创建

为创建一个单独的卷积核，方便之后建立 Vgg19 神经网络时统一调用。创建 convLayer 函数作为单独的卷积核。根据公式 2-1，为获取每一层的不断变化的卷积核的权值和偏至值。使用了 tensorflow 的内置函数 tf.get_variable。tf.nn.conv2d 是 tensorflow 内置的关于卷积神经网络的函数，它可以自动的进行输入矩阵和权值矩阵的矩阵乘法运算，我们使用它来进行卷积运算。Stride 内定义的每次移动的步长，可以自己指定。每一层使用公式 2-3 的 relu 函数进行激活，使用 tf.nn.relu 作为激活函数。代码如下：

```
def  juanjihe(x,gaodu，kuandu，buchang1，buchang2，
                Tezhengshu，name，padding = "SAME"):
    Tongdao = int(x.get_shape()[-1])
    with tf.variable_scope(name) as scope:
        a = tf.get_variable("a"，shape = [gaodu， kuandu， tongdao， tezhengshu])
         b = tf.get_variable("b"，shape = [featureNum])
         featureMap = tf.nn.conv2d(x，w，strides = [1，buchang2，buchang1，1]，
         padding =  padding)
        Shuchu = tf.nn.bias_add(a，b)
         return tf.nn.relu(tf.reshape(out，featureMap.get_shape().as_list())， name =
    scope.name)
```

3.3.2 池化核与全连接层的创建

每个池化层核的的参数设置也是类似的，使用最大值池化的方式对上一层卷积层传来的参数进行池化操作，需要用到 tf.nn.max_pool()函数，kHeight，kWidth，strideX， strideY 分别为池化核的高、宽还有步长。全连接层使用 tf.nn.xw_plus_b 对从卷积层输入的矩阵进行权值和偏至值的相加。同样的，与卷积核类似，为获取不断变化变量 w 和 b，使用 tensorflow 的内置函数 tf.get_variable。在每一层，还用到 tf.nn.relu()作为输出的激活函数。

```
def  chihuaceng(x，gaodu，kuandu，buchang1，buchang2，name，padding =
            "SAME"):
```

```
    return tf.nn.max_pool(x，ksize = [1，kHeight，kWidth，1]，
                                    strides = [1，strideX，strideY，1])
def quanlianjie(x， inputD， outputD， shiyong，mingzi)
        with tf.variable_scope(mingzi) as scope:
        a = tf.get_variable("w"，shape = [inputD， outputD]， dtype = "float")
        b = tf.get_variable("b"，[outputD]，dtype = "float")
        out = tf.nn.xw_plus_b(x，w，b，name = scope.name)
        if shiyong:
            return tf.nn.relu(shuchu)
        else:
            return shuchu
```

定义了如下函数之后，现在已经创建了单独一个卷积核，池化核和全连接层的结构了，接下来可以根据 Vgg19 的原理搭建需要的网络结构，调整卷积层，池化层的深度，甚至是全连接层的参数。

3.3.3 总体结构的搭建

具体搭建如下代码所示。首先的结构是两个卷积核接一个池化层，随后的三个结构是四个卷积层加一个池化层。根据 Vgg19 官方文档[14]，每个卷积的大小为 3x3，移动的步长为 1，从低到高的结构分别有 64，128，128，512，512 个通道，将参数传入 convLayer 函数。池化层的大小为 2x2，移动的步长为 2，将参数传入 maxPoolLayer 函数。全连接层共 4096 个权值，接受来自上一个结构传来的所有参数，根据公式 2-6。使用 softmax 函数作为目标函数，Tensorflow 可以自动计算每个核输出值的指数并最终转化成概率。

```
def  buildCNN(self):
        conv1_1 =juanjihe (self.X， 3， 3， 1， 1， 64， "conv1_1" )
        conv1_2 = juanjihe(conv1_1， 3， 3， 1， 1， 64， "conv1_2")
        pool1 =chihuahe(conv1_2， 2， 2， 2， 2， "pool1")
        conv2_1 =juanjihe(pool1， 3， 3， 1， 1， 128， "conv2_1")
        conv2_2 = juanjihe(conv2_1， 3， 3， 1， 1， 128， "conv2_2")
        pool2 = chihuahe(conv2_2， 2， 2， 2， 2， "pool2")
```

3.4 神经网络训练数据集的建立

3.4.1 数据集预处理

为了使神经网络拥有足够的性能能够提取图片中的特征信息，选用训练的数

据集同样应该足够强大包含足够多的图像，本研究用到的数据集包含 daisy，dandelion，rose，sunflower，tulip 五种花朵分类共 1079 张图片，其中 daisy 分类的一部分图片如图 3-3 所示：

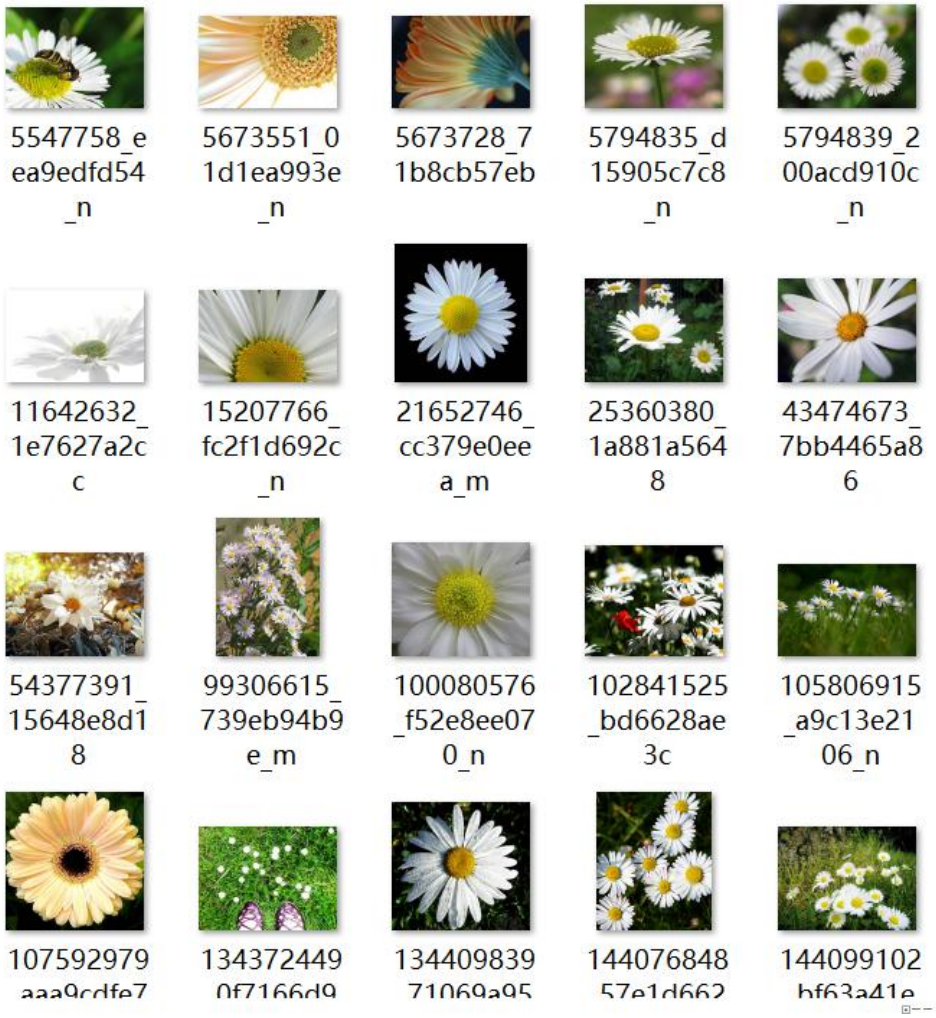

图 3-3 鲜花数据集

神经网络所需要训练的数据需要是同样大小，同样格式的。下面的程序就是对图像进行一定程度的预处理，满足训练的要求。首先使用 os 库读取保存图片的文件夹，提取后缀为.jpg、.jpeg、.png 的图片进行处理。同时将图片的尺寸统一修改为 299*299。接着，因为神经网络训练的是矩阵，也就是像素值，所以还需要将图片转换成 32 位的格式存储。代码如下：

```
for file_name in file_list:
    i += 1
    yuanshituxiang = gfile.FastGFile(mingzi，  'rb').read()
    image = tf.image.decode_jpeg(yuanshituxiang)
    if image.dtype != tf.float32:
```

```
        tuxiang = tf.image.convert_image_dtype( tuxiang ，  dtype=tf.float32)
        tuxiang = tf.image.resize_images( tuxiang ，  [299，  299])
      tuxiangzhi = sess.run(tuxiang)
```

3.4.2 图片抽取

神经网络的训练除了训练集，还要有测试集，最好的方式是能够将数据随机打乱，按一定的比例划分为训练集和测试集。为此，使用到了 np.random.shuffle()函数，此函数帮助数据集以 0.8 和 0.2 的比例划分为训练集和测试集，分别将图片数据和标签以列表的形式存储，方便以后训练的时候进行提取。因为需要处理的数据比较多，所以每次提取 200 张图片转换。将所有数据图片保存为 npy 文件为了下一步训练时使用。代码如下所示：

```
chance = np.random.randint(100)
if   chance < validation_percentage:
   validation_images.append(tuxiangzhi)
   validation_labels.append(biaoqian)
elif chance < (testing_percentage + validation_percentage):
     testing_images.append(tuxiangzhi)
     testing_labels.append(biaoqian)
else:
     training_images.append(tuxiangzhi)
     training_labels.append(biaoqian)
if   i % 200 == 0:
     print (i，  "tuxiangchuliwancheng")
```

图像预处理完之后，就可以进入神经网络训练过程。

3.5 神经网络的训练

3.5.1 模型参数的加载

首先加载之前创建的模型，提取保存的数据参数。下一步是加载 vgg 模型，使用了 slim 库快速读取，slim.getvariable()函数可以快速获取之前创建的 vgg 模型的参数，即使现在这个模型还没有训练，没有参数。为存储将来要训练的数据值，创建了一个空数组 yucun = []。

```
def   huoqucanshu():
   ex=[scope.strip()forscopeinCHECKPOINT_EXCLUDE_SCOPES.split(',')]
      yucun = []
      for goodin slim.get_model_variables():
```

```
        baohan = False
        for g in ex:
            if good.op.name.startswith(g):
                baohan = True
                break
            if not baohan:
            variables_to_restore.append(good)
    return yucun
```

3.5.2 目标函数的创建

首先使用 slim.arg_scope()函数获取上一次训练得到的全局参数。此神经网络的训练使用交叉熵作为损失，即 cross_entropy，cross_entropy 包含对数据的对数运算。对数据求导时能够得到全局最优值。tf.losses.softmax_cross_entropy()函数可以满足此要求。交叉熵损失需要数据用 one_hot 形式进行计算。one_hot 类似数组的拉平操作。把原先的二维数组拉成扁平的。还使用了 RMSPropOptimizer 针对之前定义的学习率进行全局优化求值。同时也要计算全局损失，使用 tf.losses.get_total_loss()。代码如下：

```
with slim.arg_scope(tuxiang， num_classes=N_CLASSES)
xuncans= get_trainable_variables()
 tf.losses.softmax_cross_entropy(tf.one_hot(labels，N_CLASSES)，logits)
zongsunshi = tf.losses.get_total_loss()
xunliandaishu=
tf.train.RMSPropOptimizer(LEARNING_RATE).minimize(total_loss)
with tf.name_scope('evaluation'):
        yucezhunque = tf.equal(tf.argmax(logits，1)，labels)
        daishu= tf.reduce_mean(tf.cast(correct_prediction，tf.float32))
```

每次抽取数据集中的 batch_size 个数据进行训练。将这些数据的像素值和标注类放进列表保存。每 300 代保存正确率。最后保存模型，文件保存为.npy 格式。其中包含所有层的参数。

```
for i in range(STEPS):
    _，loss = sess.run([train_step， total_loss]，feed_dict={
     images: training_images[start:end]，
    labels: training_labels[start:end]})
```

```
a.append(loss)
if i % 10 == 0 or i + 1 == STEPS:
    saver.save(sess，TRAIN_FILE， global_step=i)
    zhunquelv = sess.run(daishu，feed_dict={
    images: validation_images，labels: validation_labels})
     print('Step %d: Training loss is %.1f zhunquelv = %.1f%%' %
    b.append(zhunquelv)
    start = end
```

3.6 训练结果

除了 Vgg19 的参数之外，我们还定义了其他的参数。定义学习率为 0.0001，定义 batch_size 为 16，一次用 16 个数据进行训练，训练的代数为 3000 代。并使用 tf.train.RMSPropOptimizer()进行优化。

最后保存训练模型和参数，总共共有 1067 张图片进行训练，138 张验证数据。每 100 代使用参数测试一次验证数据集。可以看到从 400 代开始正确率上升，最开始的正确率 26.4%，训练 3000 代最后的正确率为 97.5%。训练的损失也随着迭代次数的增加而逐步减少，从 2.0 逐渐减少到 0.2。模型保存地址为'train_dir/model'。在图像风格迁移过程中可以调用。

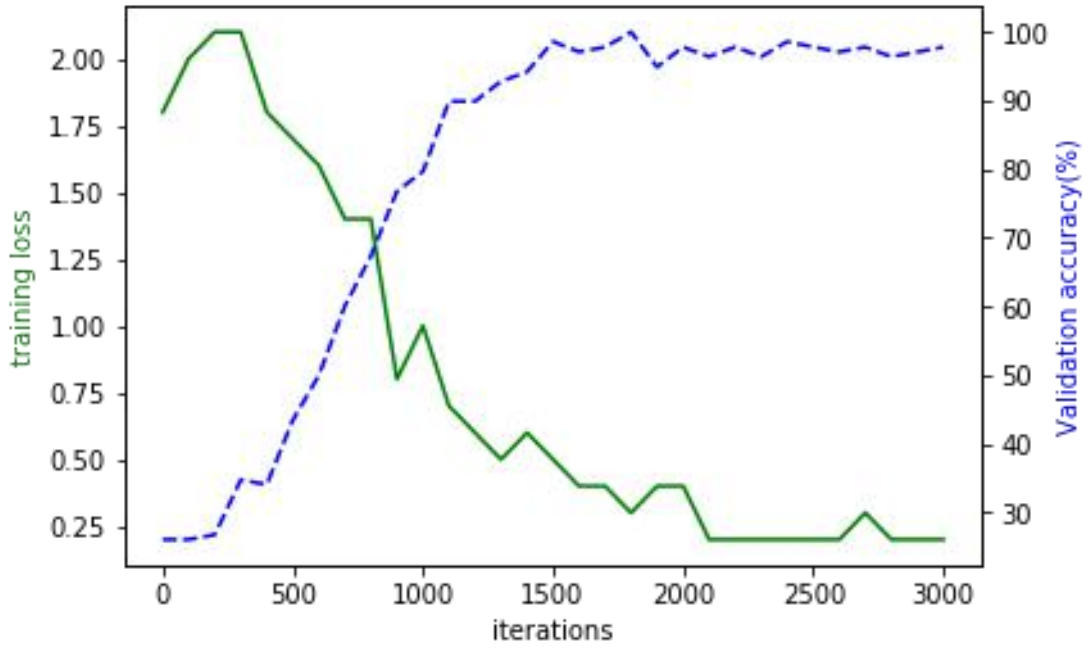

图 3-4 神经网络训练结果

3.7 小结

本章介绍了环境背景以及神经网络的训练过程，通过先对图像的处理，在进行神经网络的训练，对鲜花数据集的分类正确率能够达到 98%，意味着对图像已经拥有了不错的特征提取能力，下一章我们将介绍图像风格迁移的实现过程。

第 4 章 迁移模型的搭建与训练

本章根据前文所介绍的基于卷积神经网络的原理，迁移上文训练好的神经网络的参数，构建内容风格的损失函数，完成了风格迁移。

4.1 迁移过程

4.1.1 迁移模型的搭建

首先我们需要加载在上文中已经训练好的神经网络模型及参数，加载预训练好的 vgg 模型。用到了 keras 库，keras 是一个用 python 编写的高级神经网络 API。它可以让我们快速的进行神经网络的搭建。使用 tf.keras.applications.Vgg19 函数加载 Vgg19 模型，提取网络中的不同层，为进行前向传播过程，使用到了 vgg.get_layer.output，此函数能够计算并保存在前向传播过程当中已经出现的所有参数并锁死参数，不进行训练，只使用参数。

```
def get_Vgg19_model(layers):
    vgg = tf.keras.applications.Vgg19(include_top=False，weights='imagenet')
    outputs = [vgg.get_layer(layer).output for layer in layers]
    model = tf.keras.Model([vgg.input]，outputs)
    model.trainable = False
    return model
```

接下来定义 model 类，以便在最后训练的时候可以调用，定义 self.content_layer 和 self.style_layer 来区分提取内容特征和风格特征的神经网络层，并将它们合并之后放入到一个列表当中。使用到了字典的数据结构，创建从层名和这层输出的映射，完成这些工作后，初始化 Vgg 网络。

```
class  NeuralStyleTransferModel(tf.keras.Model):
def__init__(sel，content_layers:typing.Dict[str，float]=settings.CONTENT_LAYERS，
        super(NeuralStyleTransferModel， self).__init__()
        self.content_layers = content_layers
        self.style_layers = style_layers
        layers = list(self.content_layers.keys()) + list(self.style_layers.keys())
        self.outputs_index_map = dict(zip(layers， range(len(layers))))
        self.vgg = get_Vgg19_model(layers)
```

4.1.2 特征提取

因为需要提取特征，所以我们需要定义神经网络的前向传播过程，创建 call() 函数方法，对内容层和风格层分别进行前向传播，提取特征，保存在不同的列表

当中，首先使用到上文定义的 self.vgg 对象进行模型加载过程，这个对象可以自动的调用 keras 框架为我们提取 Vgg19 网络所有的参数，接下来我们可以根据自己的需要，根据不同的层名，从所有的参数中提取出不同层的特征参数，根据是内容层，还是特征层，加入不同的列表当中。

```
    def call(self，inputs，training=None，mask=None):
        outputs = self.vgg(inputs)
        content_outputs = []
        for layer，  factor in self.content_layers.items():
            content_outputs.append((outputs[self.outputs_index_map[layer]][0]  ，
factor))
            style_outputs = []
            for layer，  factor in self.style_layers.items():
                style_outputs.append((outputs[self.outputs_index_map[layer]][0]，
factor))
            return {'content': content_outputs，  'style': style_outputs}
```

4.1.3 全局参数设置

在创建了 Vgg19 的模型之后，为了方便后续修改参数，进行全局调整。将所有可改变的参数写进了 settings.py 函数，其中包括保存文件，保存图片的的地址，在这个函数之中，还可以设置损失函数内容层和风格层的权重。值得一体的是，在基本原理的基础之上，实现时可以规定使用 Vgg19 的哪几层提取内容或风格。甚至两种使用到的层可以不尽相同。如下代码所示：

```
CONTENT_LAYERS= {'block1_conv1': 1}
STYLE_LAYERS = {'block1_conv1': 0.5，'block2_conv1': 0.5}
CONTENT_IMAGE_PATH = './test_img/test2.jpg'
STYLE_IMAGE_PATH = './style_img/style_1.jpg'
OUTPUT_DIR = './output_img'
CONTENT_LOSS_FACTOR =1
STYLE_LOSS_FACTOR = 1000
```

4.1.4 图像预处理

我们需要减去此数据集的平均值及标准差以便归一化。用图片的 RGB 三通道值减去平均值再除以标准差作为归一化函数，同时对图像的像素进行重构，利用 tensorflow 的 tf.decode()函数进行重构，将重构后的三通道图片像素值作为训练的输入。

```
def loadmages(image_path，width=settings.WIDTH，height=settings.HEIGHT):
    x = tf.io.read_file(image_path)
    x = tf.image.decode_jpeg(x，channels=3)
    x = tf.image.resize(x， [height，width])
    x = x / 255.
    x = normalization(x)
    x = tf.reshape(x，[1，height，width，3])
    return x
```

4.1.5 内容损失函数

最初需要一张随机噪声图片，使用 np.random.uniform 在内容图片 content_image 的基础上产生一张随机噪声图片，对这张图片进行训练，目的是与风格图像贴近。根据公式 2-10，这张带有噪声的图片在每一次的训练过程中也需要经过神经网络的特征提取过程，提取出每一层的特征F_{ij}^{l}，与内容特征和风格特征进行对比。下一步首先计算内容损失，根据原理所介绍的，计算每个特征图的与内容图片对应特征P_{ij}^{l}差值的平方作为内容损失，并将其以 content_losses 列表的形式保存。

```
def compute_content_loss(noise_content_features):
    content_losses = []
    for (noise_feature， factor)， (target_feature， _) in zip(noise_content_features，
target_content_features):
        layer_content_loss = _compute_content_loss(noise_feature， target_feature)
        content_losses.append(layer_content_loss * factor)
    return tf.reduce_sum(content_losses)
```

4.1.6 风格损失函数

在创建风格层的损失函数时需要用到公式 2-11 中提到的 gram 矩阵。因为 gram 矩阵实质上是每个通道矩阵内部的像素的相互乘积，所以使用 tf.transpose() 函数变换维度，将对应的列组成一个新的矩阵，这样可以方便计算内积再使用 tf.reshape()将此计算的矩阵压缩成二维。

```
def gram_matrix(feature):
    x = tf.transpose(feature，perm=[2，0，1])
    x = tf.reshape(x，(x.shape[0]，-1))
    return x @ tf.transpose(x)
```

定义了 gram_matrix 函数之后，使用此函数分别对风格图片和现在的噪声图

片进行计算，根据公式 2-12，将每层对应的 gram 矩阵G_{ij}^{l}与风格图片A_{ij}^{l}差值的平方作为损失，计算之后加入到 style_losses 列表之中，为之后的训练做准备。此过程与内容提取过程类似。

```
def compute_style_loss(noise_style_features):
    style_losses = []
    for (noise_feature， factor)， (target_feature， _) in zip(noise_style_features，
target_style_features):
        layer_style_loss = _compute_style_loss(noise_feature， target_feature)
        style_losses.append(layer_style_loss * factor)
    return tf.reduce_sum(style_losses)
```

4.1.7 迁移网络训练

在分别计算了内容损失和风格损失之后，可以选用不同的方法对这两个损失进行训练。选择 tf.keras.optimizers.Adam()优化器进行训练。这个优化器可以打包向模型传递参数，例如学习率，衰减速率，梯度下降等等。梯度下降过程不需要手动编写，只需使用 optimizer.apply_gradients()方法即可以简单快速的优化模型，即可。

```
def train_one_step():
    with tf.GradientTape() as tape:
        noise_outputs = model(noise_image)
        loss=(total_loss(noise_outputs))
    grad = tape.gradient(loss， noise_image)
    optimizer.apply_gradients([(grad， noise_image)])
    return loss
```

4.2 风格迁移结果

4.2.1 使用不同数据集的结果

在实际的风格迁移过程中，总共拥有两个模型，一个是在我们自己的数据集上训练的 Vgg19 模型，一个是在 imagenet 官方数据集上训练的模型。两者的网络结构一样，但是由于数据集的不同，所以所训练出的神经网络的参数也不同。imagenet 数据集拥有多达 1000 个类别 30 万张图片[13]。如果要对一张不在训练数据集内的照片进行风格迁移，使用拥有更加完备的参数的 imagenet 模型势必更加有优势，但如果我们使用待风格迁移图片的数据集训练神经网络，也就意味着在这类特定的图片上提取特征的能力更强，在迁移时即使训练的数量没有 imagent，但却能呈现更好的效果

首先载入自己训练的参数，以及待风格迁移的两张图片，如图 4-1 所示：

图 4-1 左为风格右为内容

在训练过程当中，以 100 次输出一次训练图片，总共进行 20 代输出。同时使用 tqdm 库的帮助提示训练进度和每一代的损失。输出图片保存至 output 文件夹，可以看到经过每一次的迭代，损失值不断减小，20 代训练完成后，函数退出，保存图片。

```
for epoch in range(settings.EPOCHS):
    # 使用tqdm提示训练进度
    with tqdm(total=settings.STEPS_PER_EPOCH, desc='Epoch {}/{}'.format(epoch + 1, settings.EPOCHS)) as pbar:
        # 每个epoch训练settings.STEPS_PER_EPOCH次
        for step in range(settings.STEPS_PER_EPOCH):
            _loss = train_one_step()
            pbar.set_postfix({'loss': '%.4f' % float(_loss)})

            pbar.update(1)
        # 每个epoch保存一次图片
        utils.save_image(noise_image, 'output/{}.jpg'.format(epoch + 1))
```

```
train
Epoch 10/20: 100%|██████████| 100/100 [03:49<00:00,  2.29s/it, loss=5.3208]
Epoch 11/20: 100%|██████████| 100/100 [04:18<00:00,  2.59s/it, loss=5.3152]
Epoch 12/20: 100%|██████████| 100/100 [04:16<00:00,  2.56s/it, loss=5.2838]
Epoch 13/20: 100%|██████████| 100/100 [04:10<00:00,  2.50s/it, loss=5.2450]
Epoch 14/20: 100%|██████████| 100/100 [04:03<00:00,  2.44s/it, loss=5.2249]
Epoch 15/20: 100%|██████████| 100/100 [04:20<00:00,  2.61s/it, loss=5.1977]
Epoch 16/20: 100%|██████████| 100/100 [04:34<00:00,  2.75s/it, loss=5.2124]
Epoch 17/20: 100%|██████████| 100/100 [04:28<00:00,  2.69s/it, loss=5.1775]
Epoch 18/20: 100%|██████████| 100/100 [04:03<00:00,  2.44s/it, loss=5.1534]
```

图 4-2 训练结果提示

训练的损失值下降曲线如下图所示，从第 600 代开始，损失值有了明显的下降，经过 4000 次的训练，损失值由一开始的 15664 下降到了最终的 1154。

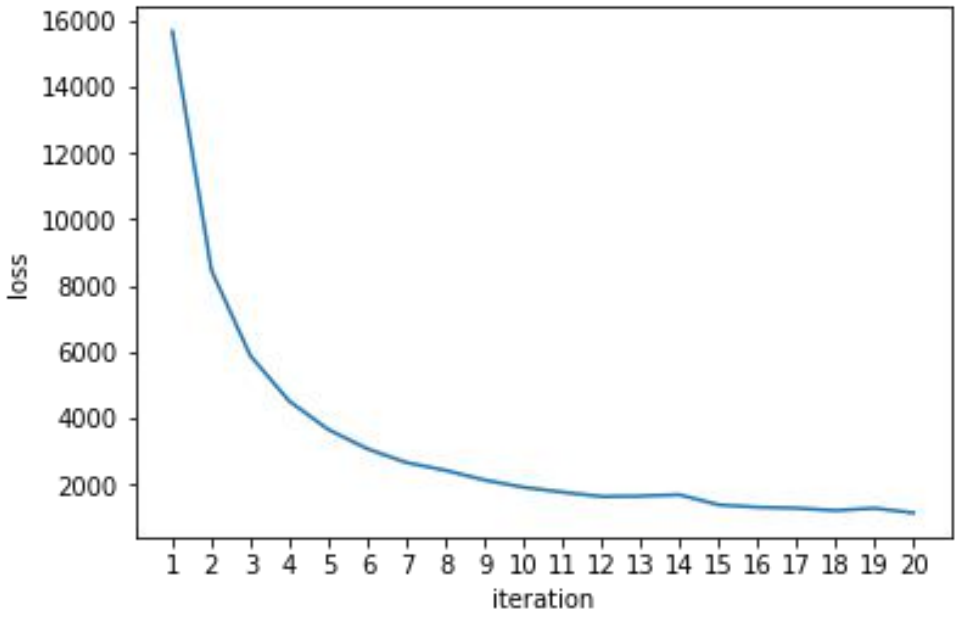

图 4-3 损失值下降曲线

被训练的内容图片在自己的数据集当中，类别标注为 daisy，为了保证提取的精度和结果对比的准确性，我们使用 Vgg19 的所有卷积层和池化层进行内容和风格的特征提取，使用同样的损失函数，训练相同的代数。使用含 daisy 的数据集的神经网络经过 20 代后的训练结果如下，图 4-4 左边为使用自己数据集的结果，右边为 imagenet 的结果：

图 4-4 两种参数的结果

从两张图片的对比可以看到，如果这张图片经过数据集的训练，那么该神经网络在提取该图像特征的时候更加精细，所以对比 imagenet，也就是这张图片不包含在数据集中的参数训练能够多提取出更加精细的特征，对内容图片的还原也就更高，也就是说，迁移后的图片在细节上能够发现更多原图的特征。

因为数据集训练的是内容图片，对风格图片没有训练，所以 imagenet 的效果

更好。在对风格图片的纹理迁移方面，表现的不够精细，使得图片更像是经过了一层滤镜的处理。

所以，如果使用自己创建的数据集训练迁移模型，在对内容图片的还原上会比直接使用参数更好，如果数据集中能够加入风格图片，也能够对其进行训练的话，将会得到更加完美的效果。在具体的使用场景中，具体的研究者可以通过实际的需要，具体情况具体分析，选择不同的参数，达到不同的效果。

4.2.2 其他参数结果

还可以通过改变风格迁移神经网络中的不同参数达到其他不同的迁移效果。在以下的所有结果对比中，都使用自己数据集的参数。

我通过改变损失函数中风格与内容的权重，可以选择弱化或者强化风格效果的表现，下面一张图中左边是风格比内容为 100:1，$\alpha = 1$，$\beta = 100$，右边是风格比内容为 100 比 1，$\alpha = 100$，$\beta = 1$，可以明显看出，两张图片的背景和纹理颜色都有很大区别，如果风格所占的权重小，那么迁移后效果并不明显，甚至可以说只是在内容照片上加了模糊化的效果，没有什么风格的特征。

同时，因为右边的图像内容占比大，而且代码中初始图片就是内容图片加噪声，所以右边的训练时间比左边快了许多，且损失值从一开始就比较小。

图 4-5 不同风格内容比的结果

更进一步的，对于风格的呈现效果，也就是图像纹理色彩的丰富性上，也可以通过增加减少提取特征的卷积池化层，或者改变这些层在风格迁移中的贡献来达到，因为神经网络的底层提取底层特征，高层提取高层特征，下面的对比中，风格与内容的比值都为 100:1。左边的图使用所有的卷积和池化层来提取特征，在对应的代码上，是{'block1_conv1'，'block2_conv1'，'block3_conv1'， 'block4_conv1'，'block5_conv1'}。

右边只使用了最高的两层，即{'block4_conv1'，'block5_conv1'}，效果的差别也很明显，使用全部参数的特征明显更丰富，体现在花朵的花蕊和背景的多彩性上，明显这些部位的色彩更加丰富，层次更加鲜明，对内容图片的细节还原也更

好。相反只使用高层参数的细节则显得比较粗犷，背景的颜色总体上更接近内容图片而不是风格，层次也没有那么丰富。

图 4-6 不同特征提取层的结果

除此之外，以上三种结果每个特征提取层的权重都是相等的，现在改变权重，结果如下图所示，对于风格和内容，依旧采用所有层提取特征，但左边图的权重为{'block1_conv1':0.1，'block2_conv1':0.1，'block3_conv1':0.2，'block4_conv1':0.3，'block5_conv1': 0.3}。即高层特征的占比大一些，而右图的权重为{'block1_conv1':0.3，'block2_conv1':0.3，'block3_conv1':0.2，'block4_conv1':0.1，'block5_conv1':0.1}.即底层的权重大一些。

图 4-7 不同特征权重的结果

可以看出，底层高权重的提取比低权重的提取背景颜色明显暗一些，而在花瓣上的纹理则更多一些。这是因为根据 2.2.1 节的原理，底层提取的更多是纹理细节，而高层更多的是人们能够分辨的颜色，形状等细节。

还有其他的很多参数可以改变，研究者可以根据具体的需要，根据风格效果呈现的需要，甚至是根据训练时间的长短，调整不同的参数，达到不同的效果。

4.3 小结

在成功训练了神经网络之后，根据基本原理搭建了对应的损失函数，并重新

定义了风格迁移的训练过程，最后通过对比两种不同的参数所呈现出的结果，说明了不同数据集训练的神经网络有不同的优势，并对其他参数也作出改变，对比了合成效果。

第 5 章 结论与展望

本文介绍了图像风格迁移的背景与过往的研究情况。介绍了基于卷积神经网络的图像风格迁移原理。设置了不同的损失函数，利用技术工具完成了迁移工作。

5.1 结论

本文从研究上来看，完成了如下工作：

（1）介绍了传统卷积神经网络的原理。并且基于传统神经网络的架构，介绍了去除原本用于图像分类的全连接层，只使用神经网络中间的卷积层和池化层参数进行特征提取的基于卷积神经网络的图像风格迁移原理，使用 Leon Gatys 创建的内容和风格的损失函数，使用随机梯度下降的方法，完成任意风格的迁移。

（2）在代码实现时，基于 Vgg19 网络，构建了卷积核池化核和全连接层，并且在鲜花数据集上完成了神经网络的训练，达到了 97.5%的准确率、拥有了提取特征的能力。

（3）在风格迁移阶段，使用神经网络的参数，依照原理构建合适的内容与风格损失函数，使用梯度下降的方式进行训练，在 3000 次训练之后得到了训练结果，完成了任意风格的迁移。

（4）对比了不同的神经网络参数之后，证明了在对某类内容图片进行风格迁移时，预先使用此类图片的数据集训练是有必要的。并且对比了损失函数中不同风格与内容的权重的迁移结果，结果表明，不同的权重组合可以实现不同的风格迁移效果。

5.2 展望

（1）因为受限于数据集，不能找到足够的风格图片进行训练，所以在使用自己的数据集时，其中只包含内容数据，这也导致了自己的神经网络在提取内容时做的足够好，在提取风格时却损失了部分细节。一个优秀的针对风格迁移的网络，其训练的数据集应该既包含内容，也包含风格。希望在以后的研究中，研究者们也能够找到方法将风格图片也加入训练集在特征提取时能够提取更多的风格。

（2）因为受限于网络的结构，损失函数的设置和当前技术的发展，不能迁移特定的风格。当下只能迁移背景，颜色鲜明强烈的风格。而对实际物体进行扭曲，重组再造的风格无能为力。这是由于卷积神经网络的中间层本质上做的是特征提取的工作，损失函数的设置也是为了去还原风格当中的颜色细节和笔触。对于艺术家的不同流派，例如毕加索的抽象主义，会产生图片扭曲失真等众多问题。

希望图像风格迁移在更多的领域能够被应用，对于不同艺术家的不同流派，

风格迁移也能很好地完成。现在风格迁移大部分还是集中在动画渲染方面，希望随着方法的改进，它能够被应用在自由创作，服装设计等更多方面，更好地改善人们的生产生活。

参考文献

[1] 陈淮源，张广驰，陈高，周清峰.基于深度学习的图像风格迁移研究进展[J/OL]．计算机工程与应用:1-11[202 1-06-05].

[2] 蔡兴泉，郭天航.面向手机应用的图像色彩风格迁移系统设计与实现[J]．信息通信，2016(06):139-140.

[3] 李兴华．线条画的提取与风格转换方法研究[D]．曲阜：曲阜师范大学，2008.

[4] 乔丽莎. 基于深度学习的图像风格艺术化[D]．西安：西安理工大学，2018.

[5] Morid M A ,Borjali AFiol G D. A scoping review of transfer learning research on medical image analysis using ImageNet[J].Computers in Biology and Medicine.2021(128):104-115.

[6] 赵杨，袁国武，谢党恩，徐丹．基于前景目标提取的图像风格化绘制算法[J]．系统仿真学报，2016，28(08)：1764-1768.

[7] Lecun Y,Bottou L. Gradient-based learning applied to document recognition[J]. Proceedings of the IEEE, 1998, 86(11):2278-2324.

[8] Krizhevsky A,Sutskever I,Hinton G E. System and method for generating training cases for image classification,US20140177947[P]. 2014.

[9] He K,Zhang X,Ren S,et al. Deep Residual Learning for Image Recognition[J].IEEE, 2016.IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016): 770-778.

[10] 卢宏涛，张秦川．深度卷积神经网络在计算机视觉中的应用研究综述[J]．数据采集与处理，2016(1)：1-17.

[11] CVPR Workshop最佳论文奖2D照片到三维人脸转化技术[J].中国教育网络，2020(08):41.

[12] 陈良. 图片风格融合及快速迁移[J]．软件工程，2021, 24(01)：21-25.

[13] 侯玉兵. 图像风格迁移方法研究[J]．中国新通信，2020，22(17)：134-135.

[14] Barlow H B,Barlow H B. David Hubel and Torsten Wiesel: Their contributions towards understanding the primary visual cortex[J]. Trends in Neurosciences, 1982, 5(82):145-152.

[15] Gorach T ,H Choksi, Kriplani M, et al. A Review on Neural Style Transfer with Auto Text Generation[J]. Social Science Electronic Publishing,2017, 33(8):1846-1869.

[16] 牟晋娟．基于深度学习的图像风格迁移技术的研究[J]．电子元器件与信息技术，2019(04)：82-85.

[17] Simonyan K, Zisserman A. Very Deep Convolutional Networks for Large-Scale Image Recognition[J]. Computer Science, 2014:1106–1114.

[18] 庞惟允．基于深度学习的图像风格迁移优化方法研究[D]．成都：电子科技大学，2020.

[19] 李杉，许新征．基于双角度并行剪枝的 VGG16 优化方法[J/OL]．计算机科学：1-12[2021

-02-14]. http://kns.cnki.net/kcms/detail/50.1075.TP.20210209.1412.045.html.

[20] 李炳臻，刘克，顾佼佼，姜文志. 卷积神经网络研究综述[J]. 计算机时代，2021(04):8-12+17.

[21] Gatys L A, Ecker A S,Bethge M. Image Style Transfer Using Convolutional Neural Networks[C]// 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2016: 2414-2423.

[22] Zhu J Y ,Park T , Isola P , et al. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks[J]. 2017 IEEE International Conference on Computer Vision (ICCV), 2017: 2242-2251

[23] 刘明昊．基于 VGG-16 的图像风格迁移[J]．电子制作，2020(12)：52-54.

[24] Barlow H B,Barlow H B. David Hubel and Torsten Wiesel: Their contributions towards understanding the primary visual cortex[J]. Trends in Neurosciences, 1982, 5(82):145-152.

[25] Gatys L A,Ecker A S,Bethge M. A Neural Algorithm of Artistic Style[J]. Journal of Vision, 2015:1097–1105.

[26] 郑泽宇，梁博文，顾思宇．Tensorflow:实战 Google 深度学习框架[M]．北京：电子工业出版社．2018. 2.

致　谢

又是一年柳絮飘扬，草长莺飞。每年的六月，伴随着燥热的空气和耀眼的阳光，总有人要离开纯真的壁垒，带着不安和期待奔赴人生的下一站。总有人要经历毕业，转身离开，以前总以为遥不可及，可现在，转眼间，现在轮到我站在这个路口，面对求学生涯的九局下半。

从去年的 12 月开始，经过 6 个月短暂开题，搜集文献，仿真实现，再到最后不断修改论文。在毕业论文终稿提交系统上按下提交的那一刹那，我诚惶诚恐。仿佛仿真结果出现的那一天还在昨日，毕业设计暂时告一段落，我的本科时光，也即将画上句号。本科四年终于圆满，入学时的懵懂激动，大三时的紧张不安历历在目，一路上跌跌撞撞，犹如大海上的孤舟，也会被突如其来的小浪所拦住了脚步，但也能在无数的艰难险阻下，一路披荆斩棘，抵达彼岸。一路上，导师如无边海洋上的一盏盏的灯塔，在我迷茫的时候，总能及时的为我照亮前方的方向，即使那光不亮，但对身处迷途之中的人来说，却格外耀眼。特别需要感谢我的毕业论文导师，虽然身体抱恙，但依然亲力亲为，从开题一直到论文提交，不厌其烦，大到仿真结果，小到格式错误，在最需要时出现，没有一句责怪。除此之外，还要感谢我的朋友和家人的照顾，他们和我一起在这片大海上航行。纵使我们每个人都是一叶孤单的小舟，有了你们的陪伴，我们便可以无惧艰险，有了你们的陪伴，我们便是世上最伟大的舰队。

我的家人们吃过许多的苦，把我送进知识的殿堂。我也走过许多的路，最终将这篇毕业论文送到导师的面前。虽然一路上磕磕绊绊，虽然其中的观点可能略显幼稚，可能在许多的方面还有欠缺，但 6 个月的时光没有白费。希望在未来的求学，工作生涯中，当我再次回顾这段本科生涯时，可以骄傲地说出不负韶华。最后正如鲁迅所说，希望今后的人生摆脱冷气，只是向上走。在过去与未来之际，愿所有的友与仇，爱与不爱者，能让我的今后更加圆满。