引用格式：毕慧敏，金鑫，周子寅，李忠兰.历史图像视频智能着色系统设计与实现[J].中国传媒大学学报（自然科学版），2022，29(06)：29-35.
文章编号：1673-4793(2022)06-0029-07

历史图像视频智能着色系统设计与实现

毕慧敏，金鑫*，周子寅，李忠兰
（北京电子科技学院，北京 100070）

摘要：为实现历史图像和视频彩色化问题，将合理、正确的颜色添加到灰度图像和视频中，利用着色评价指标对灰度图像视频着色算法进行分析，计算并分析神经网络的着色性能。设计并实现一个历史图像视频智能着色系统，前端以微信小程序为应用设计的前端展示界面，后端以PHP-Laravel-MySQL为架构的服务器和数据库框架，将着色算法与后端相连接，接收上传图像进行自动着色处理，接收视频进行分帧处理，对多帧着色再拼接还原，由前端显示。最终实现一个具有前端、后端和算法的全栈式开发架构，结合实例，证明了该系统在着色方面的有效性和便捷性。

关键词：深度学习；神经网络；图像着色；微信小程序；全栈式开发
中图分类号：TP311.1　**文献标识码：**A

Historical image video intelligent coloring software design and implementation

BI Huimin, JIN Xin*, ZHOU Ziyin, LI Zhonglan
(Beijing Electronic Science and Technology Institute, Beijing 100070, China)

Abstract: In order to realize the problem of colorization of historical images and videos, added the reasonable and correct colors to grayscale images and videos. The coloring evaluation index is used to analyze the gray-scale image video coloring algorithm, and analyze the coloring performance of the neural network by the index calculation. Design and implement a historical image video intelligent coloring system and a full-stack development architecture with front-end, back-end, and algorithms is realized finally. The front-end uses WeChat applet as the front-end display interface, and the back-end uses PHP-Laravel-MySQL as the architecture of the server and database framework. The coloring algorithm is connected to the back-end to receive uploaded images, then coloring automatically. Divided the received videosinto frames, and multiple frames are colored and then spliced and restored, and displayed by the front end. Combined with examples, it proves the effectiveness and convenience of the design in terms of coloring.

Key words: deep learning; Neural Networks;image colorization; wechat applet; full stack development

1　引言

近年来，利用计算机神经网络进行深度学习，将历史灰度图像和视频进行着色还原，成为当下人工智能和神经网络领域研究的热点之一。国内外有关自动着色的研究和应用发展迅速。例如，Richard Zhang等[1]提出采用卷积神经网络（Convolutional Neural Networks, CNN），把着色的回归任务变成一个分类任务以处理着

作者简介（*为通讯作者）：毕慧敏（1997—），女，硕士研究生，研究方向为可视计算与安全；金鑫（1983—），副教授，研究方向为可视计算与安全。Email：jinxin@besti.edu.cn

色时颜色空间的对应关系;Patricia Vitoria等[2]提出了采用生成对抗网络(GAN)模型并结合图像的语义信息对抗学习着色方法;2018年新华社联合百度AI人工智能团队开发了“给旧时光上色”的应用程序,用户上传灰度图像可返回智能着色后的结果。2020年,百度AI开放平台整合了图像特效技术,将图像着色以C++、Python、PHP等编程语言接口的形式向开发者提供,将各类图像处理技术如灰度图像着色、图像风格转换、图像增强等以接口调用形式免费提供给用户使用。但该程序对于上传图像的分辨率、大小和格式有着严格限制,在用户使用时仍存在局限性。

针对灰度图像和灰度视频在当下的实际应用需求,本文采用了HistoryNet网络结构进行进一步设计,实现了图像与视频智能着色模型的微信小程序前端,后端数据库和接口编程,利用后端进行算法远程调用,并将后端数据库进行远程部署,最终制作出基于深度学习的历史灰度图像、视频自动着色系统,前端、后端与算法分别部署,有效降低系统耦合,提高各个部分的内聚度,便于之后系统的扩展。该系统在灰度图像、灰度视频着色领域,历史档案研究和管理领域和艺术与审美等领域有较大价值。

2 着色方案设计

2.1 着色原理

着色是将合理、正确的颜色信息添加到灰度图像或灰度视频的过程。以前的黑白图像着色方法依赖于人的手工注释,并且经常产生不可称为真正着色的去饱和结果。深度学习的着色方法是利用网络中的生成器网络和判别器网络进行图像着色的“博弈”。生成器用于生成着色图像,判别器用于判定该着色图像是否真实,即是否符合人的一般认识。最终生成器进行着色生成的图像使得判别器无法区分其真实性时,即获得了一个图像着色的模型,使用该着色模型即可完成图像着色。对于灰度视频来说,可以看作是对多张连续的灰度图像着色过程,即多次灰度图像着色,因此掌握了灰度图像的着色原理,利用视频分帧和帧合成技术,就能实现灰度视频的着色。如何让计算机根据输入灰度图像对不同区域着色并返回正确结果,是深度学习自动着色领域研究的重点之一。

2.2 着色算法

在过去的十多年中,根据对神经网络的研究和改进,人们提出了几种着色技术,可分为以下三类:基于涂鸦的方法、基于参考的方法和基于深度学习的方法。其中,前两类基于用户指导,第三个基于机器学习。对于基于涂鸦的方法,需要在待着色的灰度图像上通过手工提供大量的涂鸦,费时费力;对于基于参考的方法来说,Deep Exemplar-based Colorization[3]论文第一次提出该方法,对比基于涂鸦的方法,该方法减少了人为干预的工作量,可以根据选择的不同参考图像对灰度图像进行着色。然而着色质量很大程度上受到参考图像影响,着色的结构一致性和可控性[4]把握度不强,因此在选择一幅质量较差的参考图像作为样例的时候,着色效果明显不好。若需要提高着色正确率,则需要列出参考样例可能的所有结果,必须使用海量的样例数据,这样就限制了该方法的着色效果。随着深度学习的进步,许多研究使用CNN或GAN[5]来提取灰度图片的信息以用于着色,这些方法往往实现了自然颜色匹配,但忽略了颜色分布的客观事实。

本设计以HistoryNet神经网络架构[6]实现图像和视频的着色功能,该网络包含分类、细粒度语义解析和着色三个子网络,其中分类子网络负责将图像按照年代、民族、服装类型等进行分类;语义解析子网络负责解析图像中的人物轮廓、服装和背景,生成语义分割帮助服装和人物更准确地着色,并有效防止颜色溢出;在训练过程中,将分类和语义解析融入到颜色生成网络中,以此改善着色效果。结果显示了该方法能够实现逼真的图像着色,尤其在一些语义信息比较明确的图像,特别是军装和历史图像上,该网络的处理效果是比较好的,颜色更加接近人们的常规认知。通过相关的定性和定量实验比较,本文方法在PSNR、SSIM等方面都是效果最先进的着色网络。

2.3 着色评价指标

对于一幅图像的着色来说,评价其着色好坏的标准可以分为定性评价和定量评价两方面。

定性评价,可以看作一种主观评价,是一种从人的认知角度对彩色图像的评价。定量评价[7],指的是利用着色领域内常用的指标对图像进行打分评价,主要通过PSNR(Peak Signal to Noise Ratio,峰值信噪比)和SSIM(Structural Similarity Index Measure,结构相似性)判断着色效果的好坏。

(1)PSNR是一个工程术语,表示信号最大可能功率(峰值信号)与影响其表示精度的破坏性噪声功率之比。由于许多信号都有非常广的动态范围,计算数值很大。故峰值信噪比通常取对数,用分贝(dB)来表示。

计算方法如下：

$$PSNR = 10*\log\frac{MAX_I^2}{MSE} = 20*\frac{MAX_I^2}{\sqrt{MSE}} \quad (1)$$

MAX_I表示单个图像点颜色数量的最大值。MSE是真实值与预测值之差的平方再求算术平均值，根据上式，MSE越小，则表示预测值与真实值越接近，此时PSNR值越大；MSE越大则表示预测值与真实值差距越大，此时PSNR值越小。故判断着色方法的好坏时，可以将原有的彩色图像经过灰度化之后对其进行着色，着色图与原彩色图计算PSNR值，值越大表示效果越好，着色更理想。根据统计分析列出PSNR值对应的图像着色好坏的判断指标，如表1所示：

表1 PSNR值与图像质量评价对应关系

PSNR值（dB）	图像质量评价
>40	极好
30~40	较好
20~30	较差
<20	差（不可接受）

（2）SSIM：SSIM[8]是一种衡量两幅图像相似度的指标。SSIM的度量方式：从亮度（Luminous）、对比度（Contrast）和结构（Structure）三方面度量图像相似性。

$$L(X,Y) = \frac{2\mu_X\mu_Y + C_1}{\mu_X^2 + \mu_Y^2 + C_1} \quad (2)$$

$$C(X,Y) = \frac{2\sigma_X\sigma_Y + C_2}{\sigma_X^2 + \sigma_Y^2 + C_2} \quad (3)$$

$$S(X,Y) = \frac{\sigma_{XY} + C_3}{\sigma_X\sigma_Y + C_3} \quad (4)$$

在图像对（X, Y）中，μ_X和μ_Y分别表示图像X和Y的均值，σ_X和σ_Y分别表示图像X和Y的方差，σ_{XY}表示图像X和Y的协方差（图像的数字特征计算以全图像素点为对象整体进行计算）。

$$SSIM(X, Y)=L(X, Y)*C(X, Y)*S(X, Y) \quad (5)$$

SSIM值越接近1，表示图像相似程度越高，即着色图像跟原图之间的差距越小，相应的着色效果也越好，反之则越差。

2.4 着色评价示例

结合上述着色评价指标，对于历史人物照进行着色评价，图1中人物图像是算法测试样例中综合PSNR值和SSIM值，评价最高的一组图像。直觉上看出右边的着色图和左边原图差距非常小，本文算法对人物军装和皮肤的着色非常逼真，符合历史时期的真实状况。

图1 着色示意图

（从左至右为：原图，彩色图像灰度化结果，着色结果）
着色图：PSNR=30.638 SSIM=0.962

3 系统设计与实现

3.1 系统整体架构设计

如图2所示，前端为用户操作的微信小程序，设计常用的按钮、界面；后端为用户“不可见”的数据库管理和路由管理部分，需要设计与前端以及着色算法处理相关的路由API和控制器方法，设计数据库并能够将数据通过接口以json格式返回并显示；算法部分除了需要完成相应的图像着色，还需要根据视频着色需求做出适当调整，完成资源下载、上传等。

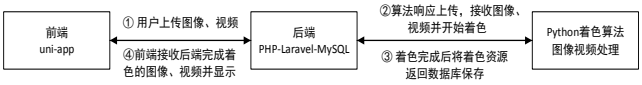

图2 系统示意图

3.2 前端开发设计

如图3所示，前端部分主要采用首页欢迎页面、图像着色页面、视频着色页面，其中包括图像、文字、按钮等编排设计和显示。

图3 前端小程序页面示意图

欢迎页面做出适当设计,以轮播图展示作为小特效,同时在下方作为上传图像的显示页面。轮播图部分采用图像滚动播放,间隔为2s一张,其中的图像左右拼接而成,左边为灰度图像,右边为彩色图像。

该部分使用GET方法,需要后端提供接口从数据库中以json格式读出作为图像路径地址在前端进行显示。显示时候将回传的json格式数据中url字段单独提取出来,可以通过IP地址、端口号和该url字段拼接为完整的图像访问地址,在界面部分即可显示出来。同时显示的图像加入点击跳转功能,即点击上传的灰度图像后,进入新页面展示。在其中go(image_id)方法中增加了参数传递功能。将图像id作为参数传递给跳转到的页面,即跳转到的页面是固定的,在跳转页面上显示的内容需要根据点击的图像不同进行不同的显示。在跳转到的页面中使用uni.request方法向数据库发送get请求,得到图像url并显示,方法中的data部分对应的是上述go方法中传递参数的接收端,接收点击跳转传递的参数供显示上传的原图和着色图像。

其中,封装好的uni.request方法参数设置如下表2所示:

表2 uni.request方法参数名及类型设置

参数名	类型
url(请求接口地址)	String
Data(请求数据)	Object/String/ArrayBuffer
Header(请求头)	Object(默认为GET)

3.3 后端开发实现

3.3.1 数据库设计

根据实际需求,设计中采用了MySQL数据库,现阶段暂时只需要两张表分别存储图像和视频信息,两张表字段类似,在这里选择图像表image_info进行阐述,具体字段设置如表3所示:

表3 图像着色数据库字段设置

名称	类型	含义
Image_id	Var(255)KEY	图像ID作为主码
Image_url	Var(255)	前端上传的路径
Image_color_url	Var(255)	着色返回的路径
Image_name	Var(255)	图像名称 Image_id.jpg
Update_time	Var(255)	前端上传时间
IP	Var(255)	前段上传的IP地址
Is_color	Boolean	是否着色

其中,上传图像后在数据库中新建一行,图像ID作为主码定义一个图像在数据库中作为一行信息保存,存储其上传路径供Python算法调用着色,同时设置图像Is_color字段为"0"表示此时未着色,着色完成后需要将着色图像保存,其路径保存在据库,路径根据原图像的主码定位到数据库其中一行,执行插入操作将着色完成的图像路径插入到Image_color_url字段中。其余字段可以根据需求添加。

3.3.2 前端-后端接口设计

后端接口主要包括路由设计和控制器设计。在使用PHP的Laravel框架设计中,路由文件在routes目录下给出,以PHP类的形式呈现给用户,包括api、channels、console和web四类路由。后端过程中,web的路由由于需要进行CSRF保护检查,对于前端连接和算法的连接非常不友好,甚至无法传递数据。故最后选择api路由实现路由编写。

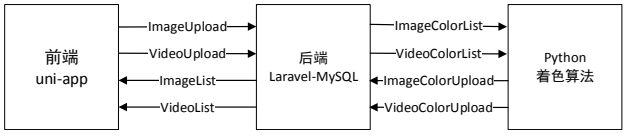

图4 后端接口API设计示意图

如图4所示,共设计8个路由,图像视频各4个,对应每一个操作分为图像(Image)和视频(Video)两类,根据上传资源是图像还是视频对应不同的路由,其中每一类路由完成操作基本相同。控制器同路由配套使用,使用控制器类函数方法作为路由请求、路由处理和路由控制等行为。控制器能够将相关逻辑组成一个单独的类。控制器被存放在app/Http/Controllers目录下。下面将以图像对应的路由和控制器为例说明各路由设计思路:

(1)ImageUpload路由:小程序端实现灰度图像的上传功能,使用POST请求,POST请求用于将数据发送到服务器来创建或更新资源,由于POST请求对数据长度和数据类型无限制,参数不会被保存在浏览器历史或web服务器日志中,数据也不会显示在URL中,因此POST适用于作为资源的上传。路由为http://IP:port//api/uploadimage。控制器代码分为两部分,一是获取上传文件,二是根据上传的操作需要将获取文件的相关字段存入数据库中。利用PHP自带的预定义数组$_FILES获取通过POST方法上传的文件并设定了存入数据库的相关字段。例如图像主码为上传名字和时间戳的MD5算法加密值,时间戳精确到秒。

确保了每一次进行MD5算法加密后的字符串完全不同，确保了主码的唯一性。

（2）ImageListController路由：显示所有上传的图像使用GET请求实现将所有上传图像显示的操作，GET请求是用于从指定资源请求数据，即请求指定的页面信息，并返回实体主体。路由地址为http://IP:port/api/imagelist。控制器部分为简单的数据库查询操作，将内容以json格式显示在页面上供前端查询使用即可。

（3）ImageListUncolorController路由：显示所有未着色图像，使用GET请求实现查询尚未着色图像操作，路由地址为http://IP:port/api/imagelist/uncolor，这里只需要使用is_color字段，查询其中值为0部分即可。

3.3.3 后端算法交互

使用Python语言编写同后端数据库交互的请求函数，然后从后端获取图像路径，将图像下载到指定文件夹，调用着色函数完成图像着色，着色完成后将着色图像上传回服务器。

（1）获取图像路径：后端已经编写好负责传递未着色图像的API接口，此时若存在未着色的图像，应该是刚刚上传上来的一张新图像。利用Python中request包下面的request.get方法访问上面的API接口，将request.get返回的json数据解码为Python对应的utf-8格式。

（2）图像下载：利用写入文件函数file.write()，将图像在服务器中的路径、图像名和后缀名提取出并写入着色函数取得图像的路径。

（3）调用着色函数完成着色：调用Python与操作系统相关的库os.system，将所给的字符串转换为指令形式在机器上运行，即可以触发指定位置的文件执行。

（4）图像上传：上传操作利用Python.request库中包含的POST方法。给出上传文件结构，包括文件路径和打开方式（以二进制只读方式打开），后端给出上传接口即可完成上传，并在后端接收到上传文件。

算法感知后端动态变化：算法部分需要自动判断是否有新的图像上传，采用最基本也是最简单的轮询（Polling）操作实现。即使用一个“死循环”while操作不断判断后端接口是否有新图像上传，若存在则进入上述的下载—着色—上传操作，若无新图像则一直等待即可。

3.4 着色流程设计

3.4.1 图像着色流程

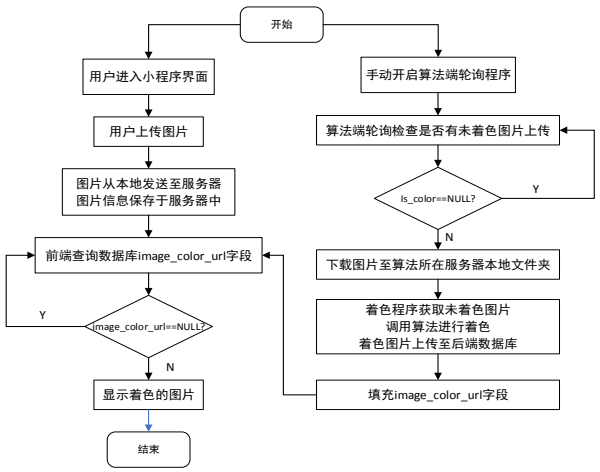

图5 图像着色算法流程图

如图5所示，着色算法设计结构中，输入和输出部分设计为读取文件夹中形式。在程序中，设算法读取DATA文件夹下所有图像，并将其作为输入进行着色完成后输出至OUT文件夹。因此设计自动着色方案时，特别增加了对文件和文件夹的操作，每一次下载一张图像，着色完成后立即将其移动到TEMP文件夹下，确保每一次着色的时候，DATA文件夹里面有且只有此时上传的图像。若不进行移动操作，程序将会对DATA文件夹下所有文件进行着色，此操作可避免浪费系统资源。

3.4.2 视频着色流程

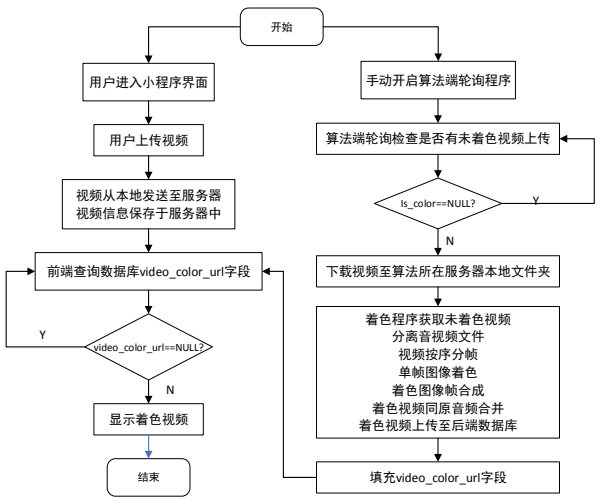

图6 视频着色算法流程图

如图6所示，视频着色原理[9]同图像着色大同小异，即视频着色相当于连续的图像着色。通过对视频进行分帧后再进行图像的着色，随后再将这些着色后图像进行合成，本文合成视频的时候进行了视频帧之

间颜色和内容的一致性处理,同时添加原音轨,确保音频和视频同步播放。

4 应用与测试

4.1 图像着色测试

在图像上传页面,选择需要着色的图片进行上传,如图7所示,上图古代女子演奏乐器时的图像,下图为劳动人民工作后的休息场面,通过对真实历史影像的着色对比后可以看出,本设计对历史人物着色效果理想,能够很好的还原人物皮肤和衣物、建筑的颜色。

图7 图像着色测试样例(左:着色前,右:着色后)

4.2 视频着色测试

(1)视频分帧操作:读取视频并获取视频帧数,按照一帧截图一张的操作,将每一张图像保存重命名并利用cv2.write函数写入指定文件夹。根据视频时长进行循环操作,需要将图像按照数字顺序保存,同时为了便于后面着色以及合成处理,图像命名时统一按照6位数字命名,不足补零。即000001.jpg,000002.jpg……以此类推。根据计算机性能不同,处理一份24帧,时长3分钟左右的视频需要裁剪大概4000~5000张图像,裁剪时间在3分钟左右。

(2)控制台显示着色过程:调用模块、计算本次着色图像数量之后开始着色。如图8所示,将视频切分为连续帧进行着色,可以看到连续着色效果较稳定,能够较好还原历史影像的状态。

图8 视频连续帧着色测试样例(上:着色前,下:着色后)

(3)视频合成操作:由于分帧、着色过程对所有图像不改变其长宽,故上述处理后所有图像大小完全相同且在文件夹中图像按照命名格式以序号排序,确保合成后的前后帧同原来相同。首先读取第一张图像作为第一帧,获取图像的长宽信息作为视频的长宽,指定编码格式(一般指定为mp4即可),根据分帧前原视频的帧数确定合成时的帧数,最后读取文件夹下所有图像,按照先后顺序将图像写入创建好的视频文件中即完成视频合成。

分帧着色过程中,若不考虑帧与帧之间的关系,只对单帧进行着色,合成视频后可能会出现由于丢失运动信息造成一定的闪烁情况,影响视频的连贯性和一致性。为此,我们在视频合成时进行帧间一致性处理[10],优化合成视频,使得着色后视频在时间稳定性和与处理帧的感知相似性之间取得平衡,提高着色后视频的连贯性和完整性。

(4)视频编码与格式转换:测试中,调用的视频编码模块,编码为内置的mpeg4文件,属性如图9所示:

图9 mpeg4编码结果

由于小程序调用HTML5页面的video插件,该内置解码器解码得到的.mp4尽管可以在本地使用播放器播放,但是不支持HTML5页面播放,故在小程序端无法播放视频。经过详细研究之后,使用Linux服务器端的FFmpeg函数操作对文件进行重新编码[11],即生成的.mp4文件重新编码,结果仍然为mp4文件,但是由于使用的编码器不同,输出的新文件可以在HTML页面播放。编码转换后属性如图10所示。

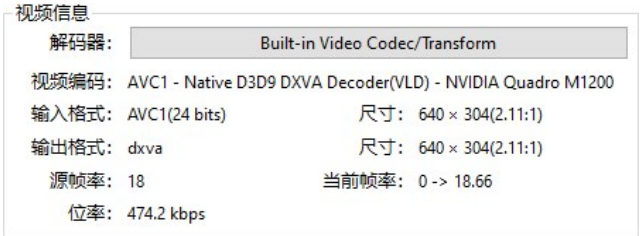

图10 H264编码转换后结果

(5)视频与音频合成:在分帧后,原视频只留下单帧

的若干张图像，音频部分（音轨）丢失。进行帧合成[12]的时候需要拼接出视频音轨。方法 是在分帧前将原视频的音频部分保存，合成帧的时候按照等帧率合成为着色后的视频并将原来保存下来的音频与合成后的视频拼接成为完整的视频文件，最后根据需要再调用视频与音频合成：在分帧后，原视频只留下单帧的若干张图像，音频部分（音轨）丢失。进行帧合成的时候需要拼接出视频音轨。方法 是在分帧前将原视频的音频部分保存，合成帧的时候按照等帧率合成为着色后的视频并将原来保存下来的音频与合成后的视频拼接成为完整的视频文件，最后根据需要再调用。

5 结论

本文利用HistoryNet算法作为历史灰度图像和视频的着色算法，在此基础上进一步设计了历史图像视频着色的系统，完成了一次全栈式开发。前端利用uni-app和微信小程序编写显示页面供用户交互，后端进行接口路由和API的编写、后端数据库设置和完善，再到前后端和算法之间利用接口进行通信连接，形成了前端、后端、算法之间接口的封装。对前端、后端、算法分别进行部署，前端和后端运行在本机，算法部署在局域网下的服务器，利用网络进行有效快速的通信以完成整个着色过程。实验测试结果表明，本文系统实现了图像和视频的自动着色功能，前端界面设计对用户操作要求低，使用便捷；前后端及算法的部署方式有利于降低系统耦合，提高各个部分的内聚度，有利于后续系统的扩展。

参考文献（References）：

[1] Zhang R, Isola P, Efros A A. Colorful image colorization [C]//European Conference on Computer Vision, 2016: 649-666.

[2] Vitoria P, Raad L, Ballester C. Chromagan: Adversarial picture colorization with semantic class distribution[C]// Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, 2020: 2445-2454.

[3] He M, Chen D, Liao J, et al. Deep exemplar-based colorization[J]. ACM Transactions on Graphics, 2018, 37(4): 1-16.

[4] Messaoud S, Forsyth D, Schwing A G. Structural consistency and controllability for diverse colorization [C]// Proceedings of the European Conference on Computer Vision (ECCV), 2018: 596-612.

[5] Alippi C, Disabato S, Roveri M. Moving convolutional neural networks to embedded systems: the alexnet and VGG-16 case[C]//17th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN), 2018: 212-223.

[6] Jin X, Li Z, Liu K, et al. Focusing on Persons: Colorizing Old Images Learning from Modern Historical Movies [C]//Proceedings of the 29th ACM International Conference on Multimedia, 2021: 1176-1184.

[7] Tong Y, Zhang Q, Qi Y. Image quality assessing by combining PSNR with SSIM [J]. Journal of Image and Graphics, 2006, 11(12): 1758-1763.

[8] Wang Z, Bovik A C, Sheikh H R, et al. Image quality assessment: from error visibility to structural similarity[J]. IEEE Transactions on Image Processing, 2004, 13(4): 600-612.

[9] Li A, Zhao S, Ma X, et al. Short-term and long-term context aggregation network for video inpainting[C]//European Conference on Computer Vision, 2020: 728-743.

[10] Lai W S, Huang J B, Wang O, et al. Learning blind video temporal consistency[C]//Proceedings of the European Conference on Computer Vision (ECCV), 2018: 170-185.

[11] Gupta M, Shah S, Salmani S. Improving whatsapp Video Statuses using FFMPEG and Software based encoding [C]//International Conference on Communication Information and Computing Technology (ICCICT), 2021: 1-6..

[12] Zeng Y, Fu J, Chao H. Learning joint spatial-temporal transformations for video inpainting [C]//European Conference on Computer Vision, 2020: 528-543.

编辑：王谦