所在组别	2022 年第二届中国高校大数据挑战赛	参赛编号
本科组		bdc221039

图像信息隐藏

摘要

互联网的快速发展，给图像、视频的传播方式带来巨大变化。图像作为媒体的重要载体，每天都有大量的而原创图像公开在互联网上。近年来，图像水印算法逐渐成为了保护图像版权的重要手段，当发生版权纠纷时可以通过算法提取嵌入的信息，从而确定图像的版权归属。本文聚焦于图像水印算法，检测图像隐藏信息[1]，并分析当前图像水印算法原始图像和嵌入水印的差异性，进行差别图像之间的特征分析和整理。并设计图像隐藏信息的检测算法，检测一张嵌入信息图片中存在的信息[2]，并对算法进行校检。

对于问题一，依据图像质量评价指标得到嵌入水印与原始图像之间的差别图像，并根据得到的差别图像进行特征提取，从而得到差别图像的共同特征。图像信息的隐藏算法的图像质量评价指标有很多，其中影响视觉效果的指标是不可见形，可用来衡量嵌入水印的图像与原始图像之间的差异性。在问题一中，本次研究通过使用多种图像质量评价指标说明原始图像与嵌入水印之后的差别，并采用的合适的统计方法说明差别图像之间的至少三种共同特征。图像质量评价指标分为主观评估和客观评估。主观评估在此问题中无意义，故不做讨论。而客观评估分别为全参考评估、半参考评估和无参考评估，为了充分利用图像信息，本报告采用的图像质量评估均为全参考评估指标，例：均方误差 MSE、平均绝对误差 MAE、归一化均方误差 NMSE、信噪比 SNR 与峰值信噪比 PSNR、结构相似性 SSIM。

对于问题二，本次研究首先正则提取有效的水印标签，然后为了后续模型的训练对水印标签进行 Acssi 码的映射，然后进行特殊的编码处理，为了减小神经网络的特征输入，对图像进行 HOG 描述子，随机选取 10%的数据作为验证集，最后使用残差网络 resnet152 和迁移学习搭建神经网络架构，对提供的 9950 张照片进行学习，并根据已得信息对准确率进行校检。

对于问题三，首先构建一个模型 model2 对图像是否嵌入水印进行判断，对于确定嵌入水印的图像使用问题二训练的模型 model1 进行具体的嵌入信息预测。然后对“附件 2”文件夹下 test_images 中包含的 50 张图片（包含嵌入和没有嵌入信息的图像）进行判断预测嵌入的水印信息。

关键词：resnet152；迁移学习；残差网络；图像质量评价指标；图像信息检测

一、 问题重述

1.1 问题背景

互联网的快速发展，给图像、视频的传播方式带来巨大变化。图像作为媒体的重要载体，每天都有大量的而原创图像公开在互联网上。通过对网络图片进行嵌入水印，可以更好的保护图片的信息版权，增强互联网信息安全。

1.2 问题提出

问题 1：图像信息隐藏算法的图像质量评价指标很多，其中影响视觉效果的 指标具有不可见性，可用来衡量嵌入水印的图像与原始图像之间的差异性。现请 你根据附件 1 中的数据，使用多种图像质量评价指标说明原始图像与嵌入水印之 后图像之间的差别，并使用合适的统计方法说明差别图像之间至少 3 种共同的特征。参考图 2 的方式展示，论文中仅需要展示“im100067. jpg” 和“im10234. jpg”两张图像的差别图像。

问题 2： 运用提供的 9950 张图片和对应的标签信息，尽量使用问题 1 中的共同特征来设计图像信息隐藏的检测算法，检测任意一张嵌入信息图片中存在的信息[3]，并对你的算法进行校验。检验量化指标为 ：9950 张图像中被正确检测信息的图像数量。

问题 3：使用检测算法对“附件 2”文件夹下 test_images 中包含 50 张图片(包含嵌入和没有嵌入信息的图像)，请先判断是否嵌入信息(信息为 a~z，A~Z，0~9 中的某一个字母或数字)。如果嵌入了信息请检测出对应信息。

二、 问题分析

2.1 问题一分析

图像信息的隐藏算法的图像质量评价指标有很多，其中影响视觉效果的指标是不可见形，可用来衡量嵌入水印的图像与原始图像之间的差异性。在问题一中，本次研究通过使用多种图像质量评价指标说明原始图像与嵌入水印之后的差别，并采用的合适的统计方法说明差别图像之间的至少三种共同特征[4]。

将问题一拆分来看，首先需要依据图像质量评价指标得到嵌入水印与原始图像间的差别图像，进而根据得到的差别特征进行特征提取，得到差别图像的共同特征。

2.2 问题二分析

在问题二中，本次研究通过运用提供的 9950 张图片和对应的标签信息，使用了问题一中的颜色、纹理、角点检测等共同特征。检测了嵌入信息图片存在的信息，并对本次研究使用的算法进行了校验。

2.3 问题三分析

在问题三中，本次研究使用相应的检测算法检测了相应的附件 2 文件夹 test_images 中包含的 50 张照片（包含嵌入和没嵌入信息的图像），首先进行判断是否嵌入信息（信息 为 a~z，A~Z，0~9 中的某一个字母或数字），对嵌入信息的图像，本次研究进行嵌入信息的判断，并将相关检测结果填写到表中。

三、 问题一模型的建立与求解

3.1 图像质量评价指标选取与说明

一般地，图像质量评价指标分为主观评估和客观评估。主观评估在此问题中无意义，故不做讨论。而客观评估分别为全参考评估、半参考评估和无参考评估，为了充分利用图像信息，本报告采用的图像质量评估均为全参考评估指标，例：均方误差 MSE、平均绝对误差 MAE、归一化均方误差 NMSE、信噪比 SNR 与峰值信噪比 PSNR、结构相似性 SSIM[5]。

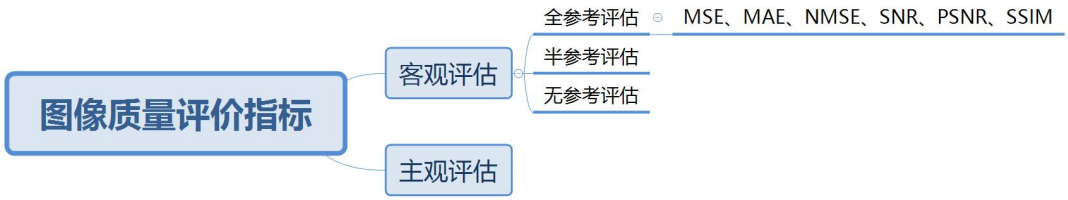


图 1 图像质量评价指标

3.1.1 均方误差 MSE 与归一化均方误差 NMSE

均方误差（mean-square error, MSE）是反映估计量与被估计量之间差异程度的一种度量。计算公式如下：

$$MSE = \frac{1}{m}\sum_{i=1}^{m}(y_i - y_i^{'})^2$$

3.1.2 平均绝对误差 MAE

MAE 是所有单个观测值与算术平均值的偏差的绝对值的平均，计算公式如下：

$$MAE = \frac{1}{m}\sum_{i=1}^{m}|y_i - y_i^{'}|$$

3.1.3 信噪比 SNR 与峰值信噪比 PSNR

PSNR 事实上只是将 MSE 换成了信号处理中常用的 db 表达形式，所以 PSNR 与 MSE 没有什么本质上的不同，只是数字上看起来更加有区分度一些。由于 PSNR 公式将 MSE 放在了分母上，所以在进行图像质量评估时，PSNR 数值越大表明待评估的图像质量越好，这一点与 MSE 相反。PSNR 公式中的 L 是一个常数，它表示图像数据类型的最大动态范围。计算公式如下：

$$PSNR = 10log_{10}\frac{(2^n-1)^2}{MSE} = 20log_{10}\frac{2^n-1}{MSE}$$

MSE 与 PSNR 的问题是，在计算每个位置上的像素差异时，其结果仅与当前位置的两个像素值有关，与其它任何位置上的像素无关。这也就是说，这种计算差异的方式仅仅将图像看成了一个个孤立的像素点，而忽略了图像内容所包含的一些视觉特征，特别是图像的局部结构信息。而图像质量的好坏极大程度上是一个主观感受，其中结构信息对人主观感受的影响非常之大。采用这种方式计算出来的差异有时不能很好地反映图像质量，所以下面将介绍 SSIM。

3.1.4 结构相似性 SSIM

SSIM(Structural SIMilarity)，结构相似性，是一种衡量两幅图像相似度的指标。

该指标首先由德州大学奥斯丁分校的图像和视频工程实验室(Laboratory for Image and Video Engineering)提出。作为结构相似性理论的实现，结构相似度指数从图像组成的角度将结构信息定义为独立于亮度、对比度的，反映场景中物体结构的属性，并将失真建模为亮度、对比度和结构三个不同因素的组合。用均值作为亮度的估计，标准差作为对比度的估计，协方差作为结构相似程度的度量。计算公式与参数说明如下：

μ_x是x的平均值，μ_x是y的平均值，σ_x^2是x的方差，σ_y^2是y的方差，σ_{xy}是x与y的协方差，$c_1=(k_1L)^2, c_2=(k_2L)^2$是用来维持稳定的常数,$k_1=0.01, k_2=0.03, L$是像素值的动态范围，一般$L=255$

$$SSIM=\frac{(2\mu_x\mu_y+c_1)(\sigma_{xy}+c_2)}{(\mu_x^2+\mu_y^2+c_1)(\sigma_x^2+\sigma_y^2+c_2)}$$

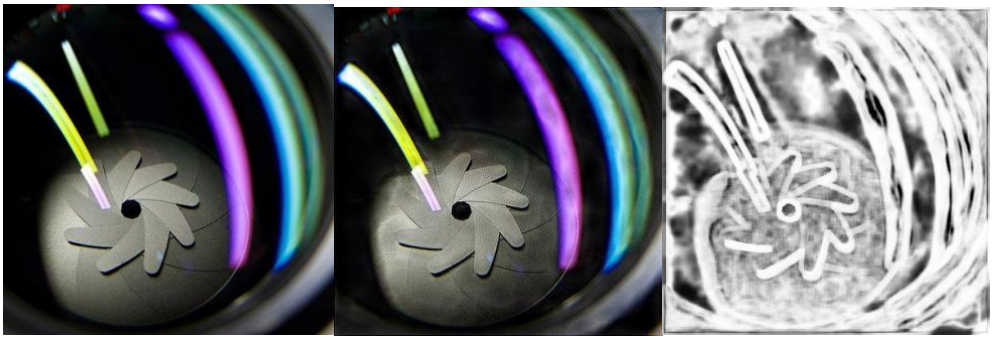
3.1.5 结果说明与展示

依据赛题要求，论文中仅需要展示“im100067. jpg”和“im10234. jpg”两张图像的差别图像。计算结果如下：

表 1 评估指标计算结果

评估指标	Im100067. jpg	Im10234. jpg
MAE	6. 989556250000000	13. 998218750000000
MSE	63. 472662500000000	1. 414647375000000
NMSE	0. 125927320669252	0. 199762428209913
SNR	8. 998800369398655	6. 994861915630120
PSNR	30. 104936447375717	26. 624321628287040
SSIM	0. 7518821325089674	0. 7906999564376284

差别图像展示如下：



Im100067. jpg　　原始图像　　嵌入水印的图像　　差别图像

Im100234. jpg　　原始图像　　嵌入水印的图像　　差别图像

3.2 共同特征

在本次研究中，根据相应的统计方法，对差别图像的共同特征：颜色特征、纹理特征、点特征进行分析。

3.2.1 颜色特征

在进行颜色特征数据分析的时候，本次研究选用颜色直方图反应图像颜色的组成分布。本次实验对差异图像寻找共同特征，将图像进行旋转变换、缩放变换、模糊变换后的颜色直方图改变不大，也考虑到图像直方图对图像的物理变换不美感，因此颜色特征可作为差异图像的共同特征[6]，其中一幅差异图的颜色直方图为：

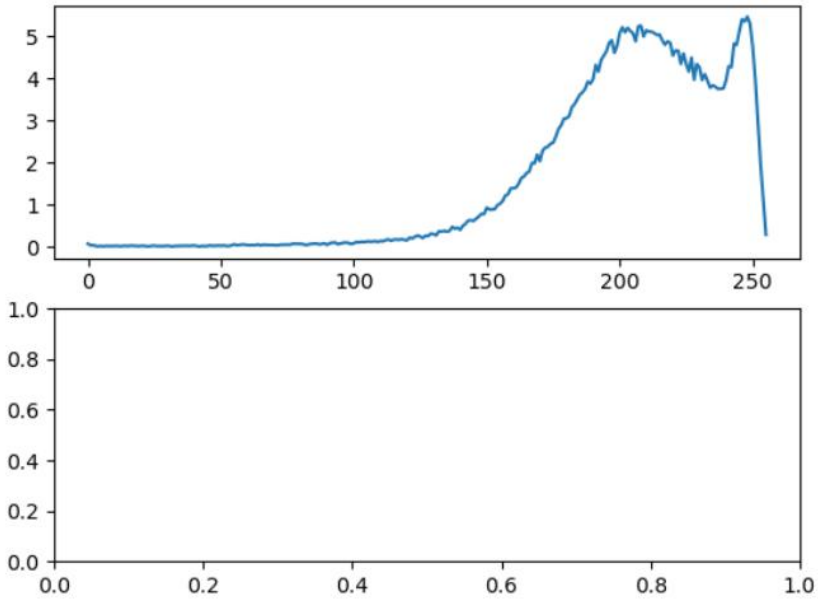


图 2 差异图的颜色直方图

3.2.2 纹理特征

图像的纹理特征是在图像计算中经过量化的图像特征。图像纹理特征可以描述图像或其中一个小块区域的空间颜色分布和光强分布[7]。因此图像的纹理特征可以作为差异图像的共同特征，其中一幅差异图的图像的纹理特征可以用下图结果展示：

```
Contrast: [[125.40685464 288.91844272 166.98796366 266.88925321]]
Dissimilarity: [[ 6.33701754 10.32108467  7.5891416   9.95133196]]
Homogeneity: [[0.2129881  0.14235249 0.18572043 0.14526407]]
Energy: [[0.02431434 0.01919781 0.02197895 0.01946374]]
Correlation: [[0.94077228 0.86349243 0.92130219 0.87391914]]
ASM: [[0.00059119 0.00036856 0.00048307 0.00037884]]
```

图 3 纹理特征结果

3.2.3 Harris 角点检测

角是图像中点似的特征，点特征不受图像平移、旋转、缩放变化的影响，对仿射变化也较为鲁棒，可作为差异图像的共同特征。其中 Harris 角点检测算法是一种基于图像灰度的方法通过计算点的曲率和梯度来检测角点[8]。下图为一幅图像的 Harris 角点检

测结果展示。

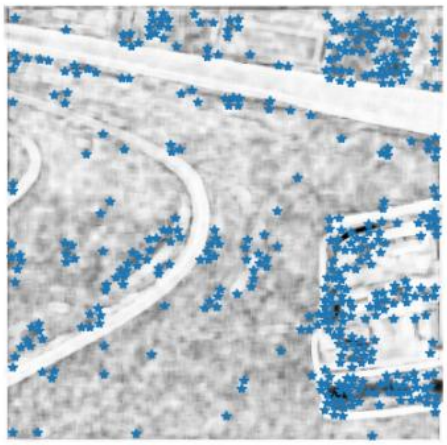


图 4 Harris 角点检测结果

四、 问题二的模型建立与求解

本次研究通过运用提供的 9950 张图片和对应的标签信息，采用残差网络和迁移学习搭建神经网络架构进行研究与学习。

4.1 数据处理

对于问题二，本次研究在数据预处理阶段主要使用了 image_message 文件夹下的图像和 message.txt 中的嵌入信息，首先对 message.txt 中的数据进行预处理，先读入数据，读入信息如下图所示：

		1	2
0	Figure Name;	Hidden Message	NaN
1		['im10001	Z']
2		['im10002	T']
3		['im10003	0']
4		['im10004	G']
9946		['im19946	j']
9947		['im19947	p']
9948		['im19948	a']
9949		['im19949	7']
9950		['im19950	6']

图 5 读入数据信息

之后，本次研究通过正则化提取有效信息，并对(a~z，A~Z，0~9)这些字符进行映射，这里采用的是映射到字符对应的 Ascii 码，例如 Z 的 Ascii 十进制是 90，下图进行举例展示：

	1	2
1	im10001	90
2	im10002	84
3	im10003	79
4	im10004	71
5	im10005	86
9946	im19946	106
9947	im19947	112
9948	im19948	97
9949	im19949	55
9950	im19950	54

图 6 Ascii 码值映射展示

然后对标签进行编码，编码规则如下

原码	运算规则	目标编码
48-57	-48	0-9
65-90	-55	10-35
97-122	-61	36-61

图 7 编码规则

编码后数据如下：

	1	2
1	im10001	35
2	im10002	29
3	im10003	24
4	im10004	16
5	im10005	31
...	...	..
9946	im19946	45
9947	im19947	51
9948	im19948	36
9949	im19949	7
9950	im19950	6

为了减少特征的输入，对图像进行了 HOG 描述子[9]处理，读取一个 batch 进行展示

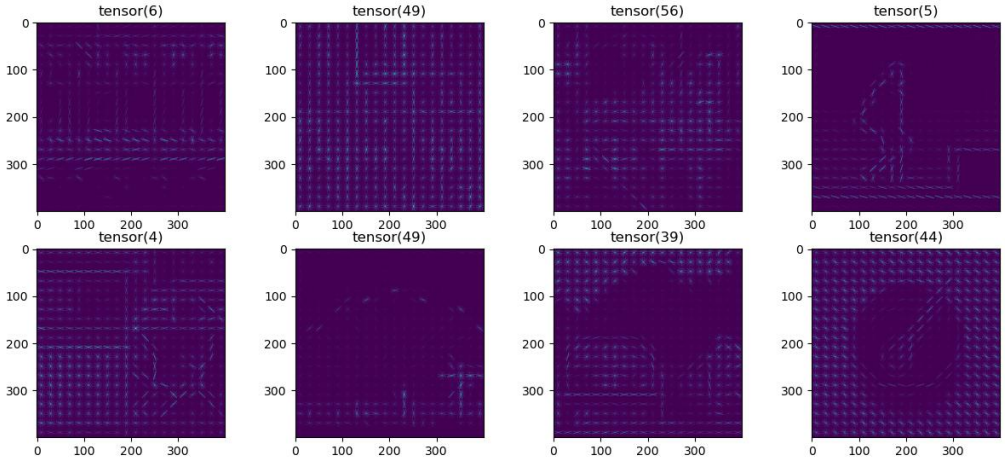


图 8 batch 结果展示

HOG 描述子实现过程如下

1、灰度化。

2、采用 Gamma 校正法对输入图像进行颜色空间的标准化，目的是调节图像的对比度，降低图像局部的阴影和光照变化所造成的影响，同时可以抑制噪音的干扰；

3、计算图像每个像素的梯度，主要是为了捕获轮廓信息，同时进一步弱化光照的干扰。

4、将图像划分成多个细胞单元

5、统计每个细胞单元的梯度直方图，获得每个细胞单元的特征描述子。

6、将每几个细胞单元组成一个块（例如 3*3 个细胞/块），一个块内所有细胞的特征描述子串联起来便得到该块的 HOG 特征描述子。

7、将图像内的所有块的 HOG 特征描述子串联起来就可以得到该图像的 HOG 特征描述子了。

梯度直方图的概念：

1、利用任意一种梯度算子，例如：sobel，laplacian 等，对一个块进行卷积，计算得到每个像素点处的梯度方向和幅值。具体公式如下：

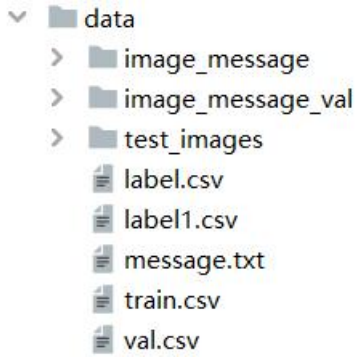
$$M(x,y) = \sqrt{I_x^2 + I_y^2} \qquad (1)$$

$$\theta(x,y) = tan^{-1}\frac{I_y}{I_x} \in [0,360^\circ) or \in [0,180^\circ) \qquad (2)$$

其中，Ix 和 Iy 代表水平和垂直方向上的梯度值，M(x,y)代表梯度的幅度值，θ (x,y)代表梯度的方向。

2、将 360 度（2*PI）根据需要分割成若干个方向，例如：分割成 6 个方向，每个 bin 包含 60 度，整个直方图包含 6 维，即 6 个方向的特征描述子。然后根据每个像素点的梯度方向，利用双线性内插法将其幅值累加到直方图中。

然后构建利用 pytorch 框架构建 mydata 数据集，并对数据集进行划分，随机取 10%的数据构建验证集，数据结构如下



其中 label.csv 为水印到 Ascii 码的映射结果，label1.csv 是对 Ascii 码进行编码后的结果，train.cv 是 image_message 对应的水印标签（编码后），val.csv 是 image_message_val 对应的水印标签（编码后）

模型的训练主要采用迁移学习，利用 resnet152 模型[10]并对后面的几层网络进行修改。普通的神经网络在层数很深时会发生效果下降的情况，而残差网络有效的避免了这一点

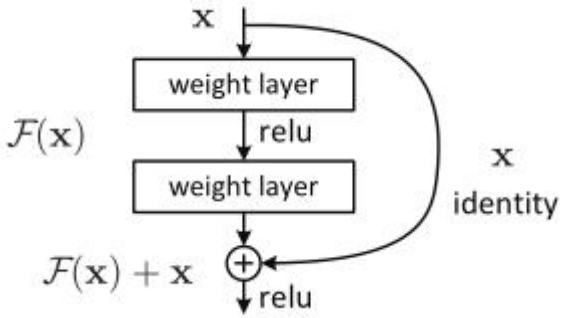


图 9：残差学习的一个构建模块

残差网络在深层的网络中表现良好

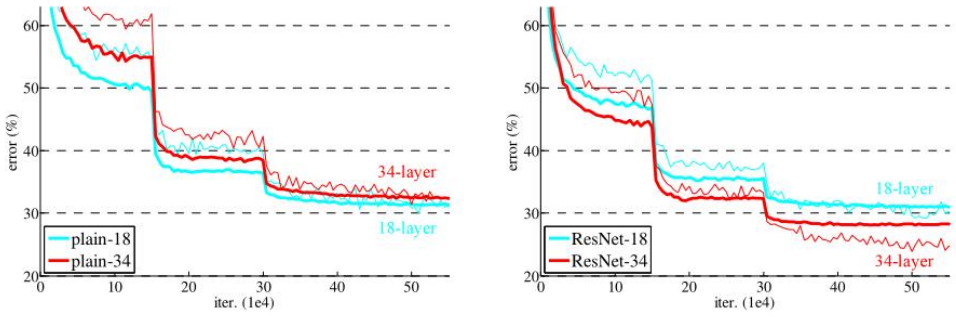


图 10：普通网络和残差网络对比图

因此这里采用了 resnet152 模型，并对最后 fc 层进行修改使之适应此问题，并且为了加速训练，模型加载了 resnet152 预训练权重参数作为模型的初始化参数构建了 model1

Model1 模型的预测结果如下（部分）

```
im19951.jpg,8
im19952.jpg,57
im19953.jpg,38
im19954.jpg,38
im19955.jpg,38
im19956.jpg,57
im19957.jpg,47
im19958.jpg,47
im19959.jpg,38
im19960.jpg,57
im19961.jpg,57
im19962.jpg,47
im19963.jpg,38
im19964.jpg,38
```

图 11： Model 1 的预测结果

4.2 信息隐藏算法检验

对图像信息隐藏算法进行检验，其正确率展示结果为：

```
train loss:2082.1986 Acc:0.8608
this is valid process
Time cost  80.0 min 9.196325063705444 s
valid loss:287.2665 Acc:0.9078
Optimizer learning rate : 0.0100000

3/3
this is train process
Time cost  114.0 min 22.266072034835815 s
train loss:2092.8605 Acc:0.8613
this is valid process
Time cost  120.0 min 0.8497114181518555 s
valid loss:378.9391 Acc:0.9022
Optimizer learning rate : 0.0100000

Training complete in 120m 1s
Best val Acc: 0.907759
```

图 12 正确率结果展示

五、 问题三的模型建立与求解

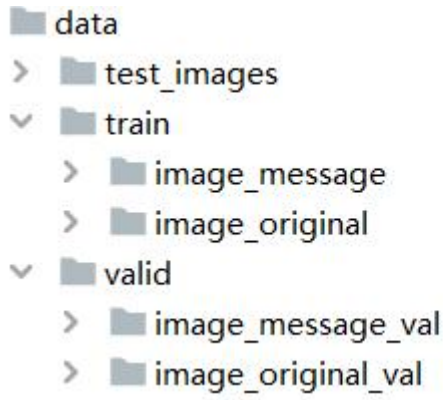
5.1 问题解析

本次研究采用问题二提出的检测算法对“附件 2”文件夹下 test_images 中包含的 50 张图片，首先进行判断是否含有嵌入信息，然后对含有嵌入信息的图片进行检验其对应的嵌入信息。对于问题三，同样采用 resnet152 进行迁移学习 model2[11]，训练数据由 'image_message'和 image_original 共同构成。

其中保存至文件中的图片与嵌入信息之间的映射关系为

{'image_message': 0, 'image_original': 1}

然后随机采用 10%的数据构成验证集，采样后文件目录结构如下



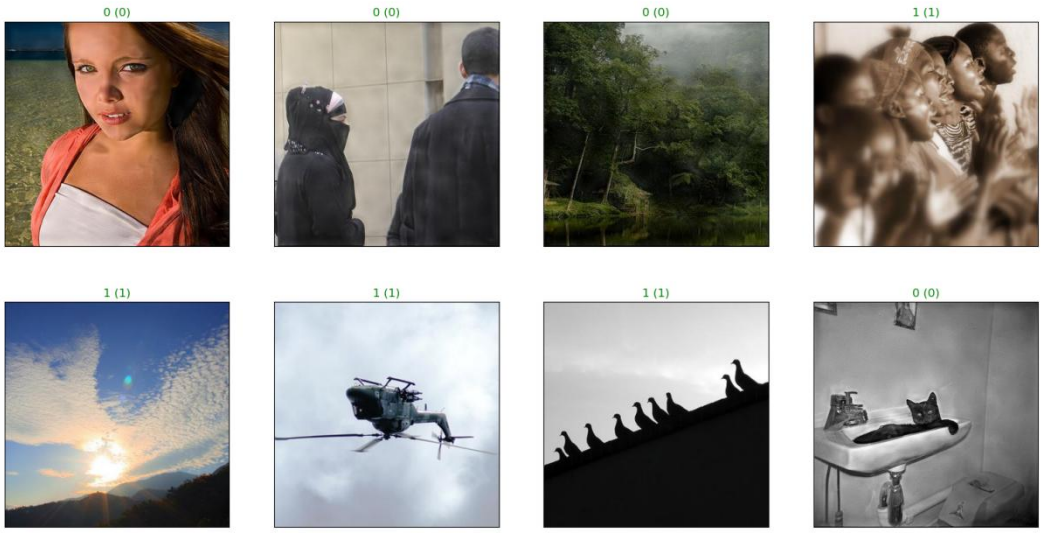
5.2 结果展示

本次研究使用的算法的检验率如下图所示

```
1/3
Time cost  47.0 min 55.69433069229126 s
train loss:856.8265 Acc:0.7717
Time cost  53.0 min 40.1102409362793 s
valid loss:78.2632 Acc:0.8382
Optimizer learning rate : 0.0100000

2/3
Time cost  108.0 min 36.50985836982727 s
train loss:868.7555 Acc:0.8110
Time cost  114.0 min 15.485408544540405 s
valid loss:117.3138 Acc:0.7945
Optimizer learning rate : 0.0100000

3/3
Time cost  168.0 min 58.498682498931885 s
train loss:952.4057 Acc:0.8081
Time cost  174.0 min 38.03833794593811 s
valid loss:48.6889 Acc:0.9025
Optimizer learning rate : 0.0100000

Training complete in 174m 38s
Best val Acc: 0.902513
```

图 13 算法准确率检验图

本次研究中，为了更好的体现本次算法的检验准确率，特列一下图表以供展示，即读取一个 valid_batch,绘图展示效果。



其中（）内的是实际标签，（）左边是预测标签，在本次实验中，这里的图片全部预测正确。准确率优秀。

本次研究中，结合 model1 和 model2 得到的预测 Ascii 值的预测截图如下：

	图像编号	嵌入的信息
0	im19951.jpg	42
1	im19952.jpg	23
2	im19953.jpg	12
3	im19954.jpg	1
4	im19955.jpg	无
5	im19956.jpg	无
6	im19957.jpg	25
7	im19958.jpg	无
8	im19959.jpg	无
9	im19960.jpg	0
10	im19961.jpg	12
11	im19962.jpg	33
12	im19963.jpg	12
13	im19964.jpg	无
14	im19965.jpg	1
15	im19966.jpg	23

图 14 预测 Ascii 值结果截图

然后对结果进行反编码，将 Ascii 值转换为最终对应的相关字符，最终结果如下：

	图像编号	嵌入的信息
0	im19951.jpg	g
1	im19952.jpg	N
2	im19953.jpg	C
3	im19954.jpg	1
4	im19955.jpg	无
5	im19956.jpg	无
6	im19957.jpg	P
7	im19958.jpg	无

图 15 预测字符结果截图

此后，本次研究将图像编号和嵌入信息的预测结果进行汇总，其中汇总图像信息如下：

图像编号	嵌入的信	图像编号	嵌入的信息	图像编号	嵌入的信息
im19951.jpg	g	im19968.jpg	1	im19985.jpg	1
im19952.jpg	N	im19969.jpg	n	im19986.jpg	C
im19953.jpg	C	im19970.jpg	X	im19987.jpg	i
im19954.jpg	1	im19971.jpg	j	im19988.jpg	P
im19955.jpg	无	im19972.jpg	无	im19989.jpg	C
im19956.jpg	无	im19973.jpg	P	im19990.jpg	g
im19957.jpg	P	im19974.jpg	C	im19991.jpg	X
im19958.jpg	无	im19975.jpg	3	im19992.jpg	X
im19959.jpg	无	im19976.jpg	i	im19993.jpg	1
im19960.jpg	0	im19977.jpg	无	im19994.jpg	1
im19961.jpg	C	im19978.jpg	P	im19995.jpg	C
im19962.jpg	X	im19979.jpg	1	im19996.jpg	U
im19963.jpg	C	im19980.jpg	1	im19997.jpg	无
im19964.jpg	无	im19981.jpg	1	im19998.jpg	X
im19965.jpg	1	im19982.jpg	6	im19999.jpg	0
im19966.jpg	N	im19983.jpg	1	im20000.jpg	1
im19967.jpg	1	im19984.jpg	1		

汇总

嵌入信息的图像总数	嵌入a~z的	嵌入A-Z的图像数量	嵌入0-9的图像数量
42	6	19	17

图 16 汇总结果展示

六、 模型评价

8.1 模型的优点

采用残差网络有效的避免了神经网络层数过多时模型效果下降的情况，并且使用 resnet152 的预训练参数作为模型的初始化参数有效的提升了模型的训练速度。

8.2 模型的缺点

因为使用了深度学习，模型的可解释性较差，特征的提取都是自动进行的，解释性比较差。

8.3 模型的改进与推广

可以在图片输入网络前通过更加复杂的特征提取算法如：BRIEF、ORB 进行特征的提取降低模型的复杂度，使模型的复杂度降低可解释性更好。

参考文献

[1]刘仁杰. 图像信息隐藏与伪装的嵌入式设计[D]. 北方工业大学, 2010.
[2]钱振兴, 吕梦琪, 黄楠楠,等. 基于纹理合成在图像中嵌入秘密信息的方法:, CN107578362A[P]. 2018.
[3]杨雪. 机器视觉中图像检测算法的研究与应用[D]. 江南大学.
[4]唐坚刚, 王泽兴. 基于 Hash 值的重复图像检测算法[J]. 计算机工程, 2009, 35(1):3.
[5]张小利, 李雄飞, 李军. 融合图像质量评价指标的相关性分析及性能评估[J]. 自动化学报, 2014, 40(002):306-315.
[6]张晓飞, 万福才, 刘朋. 主元分析法(PCA)在图像颜色特征提取中的应用[J]. 沈阳大学学报:自然科学版, 2006.
[7]王惠明, 史萍. 图像纹理特征的提取方法[J]. 中国传媒大学学报(自然科学版), 2006.
[8]赵万金, 龚声蓉, 刘纯平,等. 一种自适应的 Harris 角点检测算法[J]. 计算机工程, 2008, 34(10):4.
[9]李伟, 章军伟. 一种基于 HOG 和特征描述子的人脸检测与跟踪方法[J]. 浙江工业大学学报, 2020, 48(2):8.
[10]张黎明, 李恒, 陈金萍,等. 一种基于 ResNet 的红外与可见光图像融合方法:, CN110189286A[P]. 2019.
[11]吴田军, 骆剑承, 夏列钢,等. 迁移学习支持下的遥感影像对象级分类样本自动选择方法[J]. 测绘学报, 2014(9):9.

附录

代码名称	代码解决的问题	代码电子版所在的位置
calculate_mae.m	问题一的 MAE 计算函数	支撑材料中代码/问题一文件夹
calculate_mse.m	问题一的 MSE 计算函数	支撑材料中代码/问题一文件夹
calculate_nmse.m	问题一的 NMSE 计算函数	支撑材料中代码/问题一文件夹
calculate_snr_psnr.m	问题一的 snr、psnr 计算函数	支撑材料中代码/问题一文件夹
main.m	问题一的图像质量评价指标计算主函数	支撑材料中代码/问题一文件夹
SSIM.ipynb	问题一依据 SSIM 指标得差别图像	支撑材料中代码/问题一文件夹
color.py	问题一的颜色特征	支撑材料中代码/问题一文件夹
feature.py	问题一的 Harris 角点检测	支撑材料中代码/问题一文件夹
texture.py	问题一的纹理特征	支撑材料中代码/问题一文件夹
split_train_test.py	问题二 train 和 valid 的数据划分	支撑材料中代码/问题二文件夹
task2_model1.py	Model1 的训练和预测	支撑材料中代码/问题二文件夹
split.py	任务三 train 和 valid 数据划分	支撑材料中代码/问题三文件夹
classify.py	Model2 的训练和预测	支撑材料中代码/问题三文件夹
result.py	汇总 model1 和 model2 的结果	支撑材料中代码/问题三文件夹

calculate_mae.m

```
%该函数用于计算 MAE 评价绝对误差
%值越小表示与原始图像偏差越小，图像质量越好
%I 表示原始图像
%J 表示恢复后的图像
function mae = calculate_mae(I,J)
    %如果是 I 灰度图像只有二维，如果 I 是彩色图像将会有三维
    dim = length(size(I));%保存的是 I 的维度
```

```
    M = size(I,1);
    N = size(I,2);
    dif = abs(I - J);
    if dim == 2
        val = sum(sum(dif));
    else
        val = sum(sum(sum(dif)));
    end
    mae = val / (M*N);
end
```

calculate_mse.m

```
%该函数用于计算 MSE 计算均方误差
%mse 值越小图像质量越好。
%I 表示原始图像
%J 表示恢复后的图像
function mse = calculate_mse(I,J)
    %如果是 I 灰度图像只有二维，如果 I 是彩色图像将会有三维
    dim = length(size(I));%保存的是 I 的维度
    M = size(I,1);
    N = size(I,2);
    dif = (I - J).^2;
    if dim == 2
        val = sum(sum(dif));
    else
        val = sum(sum(sum(dif)));
    end
    mse = val / (M*N);%其实这里的 M*N=numel(I)
end
```

calculate_nmse.m

```
%该函数用于计算 MSE 计算归一化均方误差
%值越小图像质量越好
%I 表示原始图像
%J 表示恢复后的图像
function nmse = calculate_nmse(I,J)
    %如果是 I 灰度图像只有二维，如果 I 是彩色图像将会有三维
    dim = length(size(I));%保存的是 I 的维度
    dif = (I - J).^2;
    I_2 = I.^2;
    if dim == 2
```

```
        val1 = sum(sum(dif));
        val2 = sum(sum(I_2));
    else
        val1 = sum(sum(sum(dif)));
        val2 = sum(sum(sum(I_2)));
    end
    nmse = val1/val2;
end
```

calculate_snr_psnr.m

```
%该函数用于计算 SNR 信噪比和 PSNR 峰值信噪比
%信噪比与峰值信噪比的值越大代表图像的质量越好。
%I 表示原始图像
%J 表示恢复后的图像
function [snr,psnr] = calculate_snr_psnr(I,J)
    %如果是 I 灰度图像只有二维，如果 I 是彩色图像将会有三维
    dim = length(size(I));%保存的是 I 的维度
    M = size(I,1);
    N = size(I,2);
    dif = (I - J).^2;
    I_2 = I.^2;
    if dim == 2
        val1 = sum(sum(dif));
        val2 = sum(sum(I_2));
    else
        val1 = sum(sum(sum(dif)));
        val2 = sum(sum(sum(I_2)));
    end
    snr = 10*log10(val2/val1);
    psnr = 10*log10((255*255*M*N)/val1);
end
```

main.m

```
I = imread('234original.jpg');
J = imread('234message.jpg');
mae = calculate_mae(I,J)
mse = calculate_mse(I,J)
nmse = calculate_nmse(I,J)
[snr,psnr] = calculate_snr_psnr(I,J)
```

```
%isnr = calculate_isnr(I,P,J)
```

SSIM.ipynb

```
# -*- coding: utf-8 -*-
#!python -m pip install --upgrade pip -i https://pypi.douban.com/simple --user
#!pip install scikit-image
#!pip install skimage
import cv2
import numpy as np
from skimage import metrics
from scipy import signal

# -*- coding: utf-8 -*-

def compute_mse(X, Y):
    """
    compute mean square error of two images

    Parameters:
    -----------
    X, Y: numpy array
        two images data

    Returns:
    --------
    mse: float
        mean square error
    """
    X = np.float32(X)
    Y = np.float32(Y)
    mse = np.mean((X - Y) ** 2, dtype=np.float64)
    return mse


def compute_psnr(X, Y, data_range):
    """
    compute peak signal to noise ratio of two images

    Parameters:
    -----------
    X, Y: numpy array
```

```
        two images data

    Returns:
    --------
    psnr: float
        peak signal to noise ratio
    """
    mse = compute_mse(X, Y)
    psnr = 10 * np.log10((data_range ** 2) / mse)
    return psnr


def compute_ssim(X, Y, win_size=7, data_range=None):
    """
    compute structural similarity of two images

    Parameters:
    -----------
    X, Y: numpy array
        two images data
    win_size: int
        window size of image patch for computing ssim of one single position
    data_range: int or float
        maximum dynamic range of image data type

    Returns:
    --------
    mssim: float
        mean structural similarity
    ssim_map: numpy array (float)
        structural similarity map, same shape as input images
    """
    assert X.shape == Y.shape, "X, Y must have same shape"
    assert X.dtype == Y.dtype, "X, Y must have same dtype"
    assert win_size <= np.min(X.shape[0:2]), \
        "win_size should be <= shorter edge of image"
    assert win_size % 2 == 1, "win_size must be odd"
    if data_range is None:
        if 'float' in str(X.dtype):
            data_range = 1
        elif 'uint8' in str(X.dtype):
            data_range = 255
```

```
            else:
                raise ValueError(
                    'image dtype must be uint8 or float when data_range is None')

        X = np.squeeze(X)
        Y = np.squeeze(Y)
        if X.ndim == 2:
            mssim, ssim_map = _ssim_one_channel(X, Y, win_size, data_range)
        elif X.ndim == 3:
            ssim_map = np.zeros(X.shape)
            for i in range(X.shape[2]):
                _, ssim_map[:, :, i] = _ssim_one_channel(
                    X[:, :, i], Y[:, :, i], win_size, data_range)
            mssim = np.mean(ssim_map)
        else:
            raise ValueError("image dimension must be 2 or 3")
        return mssim, ssim_map


    def _ssim_one_channel(X, Y, win_size, data_range):
        """
        compute structural similarity of two single channel images

        Parameters:
        -----------
        X, Y: numpy array
            two images data
        win_size: int
            window size of image patch for computing ssim of one single position
        data_range: int or float
            maximum dynamic range of image data type

        Returns:
        --------
        mssim: float
            mean structural similarity
        ssim_map: numpy array (float)
            structural similarity map, same shape as input images
        """
        X, Y = normalize(X, Y, data_range)
        C1 = 0.01 ** 2
        C2 = 0.03 ** 2
```

```
    num = win_size ** 2
    kernel = np.ones([win_size, win_size]) / num
    mean_map_x = convolve2d(X, kernel)
    mean_map_y = convolve2d(Y, kernel)

    mean_map_xx = convolve2d(X * X, kernel)
    mean_map_yy = convolve2d(Y * Y, kernel)
    mean_map_xy = convolve2d(X * Y, kernel)

    cov_norm = num / (num - 1)
    var_x = cov_norm * (mean_map_xx - mean_map_x ** 2)
    var_y = cov_norm * (mean_map_yy - mean_map_y ** 2)
    covar_xy = cov_norm * (mean_map_xy - mean_map_x * mean_map_y)

    A1 = 2 * mean_map_x * mean_map_y + C1
    A2 = 2 * covar_xy + C2
    B1 = mean_map_x ** 2 + mean_map_y ** 2 + C1
    B2 = var_x + var_y + C2

    ssim_map = (A1 * A2) / (B1 * B2)
    mssim = np.mean(ssim_map)
    return mssim, ssim_map


def normalize(X, Y, data_range):
    """
    convert dtype of two images to float64, and then normalize them by
    data_range

    Paramters:
    ----------
    X, Y: numpy array
        two images data
    data_range: int or float
        maximum dynamic range of image data type

    Returns:
    --------
    X, Y: numpy array
        two images
    """
```

```
    X = X.astype(np.float64) / data_range
    Y = Y.astype(np.float64) / data_range
    return X, Y


def convolve2d(image, kernel):
    """
    convolve single channel image and kernel

    Parameters:
    -----------
    image: numpy array
        single channel image data
    kernel: numpy array
        kernel data

    Returns:
    --------
    result: numpy array
        image data, same shape as input image
    """
    result = signal.convolve2d(image, kernel, mode='same', boundary='fill')
    return result


def color_to_gray(image, normalization=False):
    """
    convert color image to gray image

    Parameters:
    -----------
    image: numpy array
        color image data
    normalization: bool
        whether to do nomalization

    Returns:
    --------
    image: numpy array
        gray image data
    """
    image = cv2.cvtColor(image, code=cv2.COLOR_BGR2GRAY)
```

```
    if normalization and 'uint8' in str(image.dtype):
        image = image.astype(np.float64) / 255
    return image

import os
from PIL import Image
if __name__ == '__main__':
    for filename in os.listdir(r"C:/Users/Lenovo/Desktop/image_original"):
        print(filename)
        image_original =
cv2.imread("C:/Users/Lenovo/Desktop/image_original/%s"%(filename))
        image_message =
cv2.imread("C:/Users/Lenovo/Desktop/image_message/%s"%(filename))
        psnr = compute_psnr(image_original,image_message,255)
        psnr_skimage =
metrics.peak_signal_noise_ratio(image_original,image_message,data_range =
255)
        image_original = color_to_gray(image_original,normalization = False)
        image_message = color_to_gray(image_message,normalization = False)
        mssim,ssim_map = compute_ssim(image_original,image_message)
        mssim_skimage =
metrics.structural_similarity(image_original,image_message,multichannel =
False)
        #cv2.imshow('ssim_map', ssim_map)

#ssim_map.save("C:/Users/Lenovo/Desktop/image_differential/%s"%(filename))

cv2.imwrite(r"C:/Users/Lenovo/Desktop/image_differential\{}.jpg".format(fil
ename),ssim_map*255)

#cv2.imwrite("C:/Users/Lenovo/Desktop/image_differential/%s"%(filename),ssi
m_map)
```

color.py

```
'''
颜色特征：颜色直方图
'''

import torch
from torch import nn, sigmoid
from IPython.display import Image
```

```
from IPython.core.display import HTML
import cv2
import matplotlib.pyplot as plt
import numpy as np


def np_to_torch(img, add_batch_dim=True):
    img = np.asarray(img)
    img = img.astype(np.float32).transpose((2, 0, 1))
    if add_batch_dim:
        img = img[np.newaxis, ...]
    img_torch = torch.from_numpy(img) / 255.0
    return img_torch


def phi_k(x, L, W):
    return sigmoid((x + (L / 2)) / W) - sigmoid((x - (L / 2)) / W)


def phi_step(x, L, W):
    print(x.shape)
    z1 = (x + (L / 2)) / W
    z2 = (x - (L / 2)) / W
    return sigmoid(1000 * z1) - sigmoid(1000 * z2)


def compute_pj(x, mu_k, K, L, W, kernel):
    x = x.reshape(x.size(0), 1, -1)
    x = x.repeat(1, K, 1)
    if kernel == "step":
        return phi_step(x - mu_k, L, W)
    elif kernel == "sigmoid":
        return phi_k(x - mu_k, L, W)


class SingleDimHistLayer(nn.Module):
    def __init__(self, kernel):
        super().__init__()
        self.kernel = kernel
        self.K = 256
        self.L = 1 / self.K
        self.W = self.L / 2.5
        self.mu_k = (self.L * (torch.arange(self.K) + 0.5)).view(-1, 1).cuda()
```

```
    def forward(self, x):
        N = x.size(1) * x.size(2)
        print(x.shape)

        pj = compute_pj(x, self.mu_k, self.K, self.L, self.W, self.kernel)
        return pj.sum(dim=2) / N

img_src = cv2.imread("./data1/difference/im0000.jpg")
img_src = np_to_torch(img_src).cuda()
hist_layer = SingleDimHistLayer("step")
SS = hist_layer(img_src)
_, ax = plt.subplots(2)
ax[0].plot(SS[0].cpu().numpy())
plt.show()
```

feature.py

```
'''
提取特征
Harris 角点检测算法
'''

from pylab import *
from PIL import Image
from numpy import *
from scipy.ndimage import filters


def compute_harris_response(im, sigma=3):
    imx = zeros(im.shape)
    filters.gaussian_filter(im, (sigma, sigma), (0, 1), imx)
    imy = zeros(im.shape)
    filters.gaussian_filter(im, (sigma, sigma), (1, 0), imy)

    Wxx = filters.gaussian_filter(imx * imx, sigma)
    Wxy = filters.gaussian_filter(imx * imy, sigma)
    Wyy = filters.gaussian_filter(imy * imy, sigma)

    Wdet = Wxx * Wyy - Wxy ** 2
```

```
    Wtr = Wxx + Wyy
    return Wdet / Wtr


def get_harris_points(harrisim, min_dist=10, threshold=0.1):

    corner_threshold = harrisim.max() * threshold
    harrisim_t = (harrisim > corner_threshold) * 1

    coords = array(harrisim_t.nonzero()).T

    candidate_values = [harrisim[c[0], c[1]] for c in coords]
    index = argsort(candidate_values)
    allowed_locations = zeros(harrisim.shape)
    allowed_locations[min_dist:-min_dist, min_dist:-min_dist] = 1
    # 按照 min_distance 原则，选择最佳 Harris 点
    filtered_coords = []
    for i in index:
        if allowed_locations[coords[i, 0], coords[i, 1]] == 1:
            filtered_coords.append(coords[i])
            allowed_locations[(coords[i, 0] - min_dist):(coords[i, 0] + min_dist),
            (coords[i, 1] - min_dist):(coords[i, 1] + min_dist)] = 0
    return filtered_coords


def plot_harris_points(image, filtered_coords):
    """绘制图像中检测到的角点"""
    figure()
    gray()
    imshow(image)
    plot([p[1] for p in filtered_coords], [p[0] for p in filtered_coords], '*')
    axis('off')
    show()


# 调用展示
im = array(Image.open('.\data1\difference\im0000.jpg').convert('L'))
harrisim = compute_harris_response(im)
filtered_coords = get_harris_points(harrisim, 6, 0.05)
plot_harris_points(im, filtered_coords)
```

texture.py

```
'''
纹理特征
'''

import cv2
import numpy as np

image_spot = cv2.imread('./data1/difference/im0000.jpg')
gray = cv2.cvtColor(image_spot, cv2.COLOR_BGR2GRAY)

import skimage.feature as feature

graycom = feature.greycomatrix(gray, [1], [0, np.pi /4, np.pi /2, 3* np.pi /
4], levels=256)

contrast = feature.greycoprops(graycom, 'contrast')
dissimilarity = feature.greycoprops(graycom, 'dissimilarity')
homogeneity = feature.greycoprops(graycom, 'homogeneity')
energy = feature.greycoprops(graycom, 'energy')
correlation = feature.greycoprops(graycom, 'correlation')
ASM = feature.greycoprops(graycom, 'ASM')

print("Contrast: {}".format(contrast))
print("Dissimilarity: {}".format(dissimilarity))
print("Homogeneity: {}".format(homogeneity))
print("Energy: {}".format(energy))
print("Correlation: {}".format(correlation))
print("ASM: {}".format(ASM))
```

split_train_test.py

```
# -*- coding:utf-8 -*-
import os
import random
import re
import sys
import shutil
import pandas as pd
# df=pd.read_csv('../data/label1.csv')
#
# source_path="../data/image_message"
# de_path="../data/image_message_val"
# if not os.path.exists(de_path):
#     os.makedirs(de_path)
```

```
#     print('创建'+de_path)
# val_percent=0.1
#
# filename=os.listdir(source_path)
# num=len(filename)
# val=int(num*val_percent)
# file=range(num)
# print(file)
# file_select=random.sample(file,val)
# df_val=df.iloc[file_select,:]
# df_val.to_csv("../data/val.csv",index=None)
# df_train=df.copy()
# df_train.drop(file_select,inplace=True)
# df_train.to_csv("../data/train.csv",index=None)
# print("运行到行号: ",sys._getframe().f_lineno)
# for i in file_select:
#
#     val_name=filename[i]
#     file_name_source=source_path+'/'+val_name   # 源文件路径
#     val_path_new=de_path+'/'+val_name          # 目标文件路径
#     shutil.move(file_name_source,val_path_new)
# print("结束 运行到行号: ",sys._getframe().f_lineno)
"""
val.csv 排序
"""
data=pd.read_csv('../data/val.csv')
data[3]=data.iloc[:,0].apply(lambda x:re.findall('\d+',x)[0])
data=data.sort_values(by=3)
data=data.drop(3,axis=1) # 列
print(data)
data.to_csv("../data/val.csv",index=None)
# re.findall('\d+', 'im10603') # ['10603']
```

task2_model1.py

```
# -*- coding:utf-8 -*-
import pandas as pd
import numpy as np
import os
import time
import copy
import random
import sys
import shutil
```

```
from PIL import Image
import torch
import torch.optim as optim
import torch.nn as nn
import torchvision
from torch.utils.data import Dataset,DataLoader,TensorDataset
from torchvision import transforms,models
import matplotlib.pyplot as plt

print(torch.cuda.is_available())
"""
读入标签
"""
df=pd.read_csv('../data/message.txt',encoding='gbk',header=None,names=[1,2])
print(df.head(5).append(df.tail(5)))
df.info()
df=df.dropna()

df[1]=df[1].apply(lambda x:x.replace("['",""))
df[2]=df[2].apply(lambda x:x.replace("']","").replace(' ',""))

# df['length']=df[2].apply(lambda x:len(x))   # 都是2说明有空格符号

df[2]=df[2].apply(lambda x:ord(x))
print(df.head(5).append(df.tail(5)))
df.info()
df[2]=df[2].apply(lambda x:x-48 if (x>=48 and x<=57) else x)
df[2]=df[2].apply(lambda x:x-55 if (x>=65 and x<=90) else x)
df[2]=df[2].apply(lambda x:x-61 if (x>=97 and x<=122) else x)
print(df)
# df.to_csv(r'../data/label1.csv',index=None)

"""
训练模型
"""
# 查看一下图片的形状
img1=Image.open('../data/image_message/im10001.jpg')
img1.size
img1=np.array(img1)
print(img1.shape)  # (400, 400, 3)
# plt.imshow(img1)
# plt.show()
"""
```

划分训练集和测试集

```
"""
# split_train_test



# 图片归一化标准化
transform=transforms.Compose([transforms.ToTensor(),
                              transforms.Resize((400,400)),
                              transforms.Normalize([0.485, 0.456, 0.406],[0.229,
0.224, 0.225])
                              ])


class Mydata(Dataset):
    def __init__(self,path,label_path):
        self.path=path
        self.path_list=os.listdir(self.path)
        self.path_label=label_path
        if self.path_label is not None:
            df = pd.read_csv(self.path_label)
            self.y = list(df.iloc[:, 1])


    def __getitem__(self, item):
        names=self.path_list[item]
        img_name=os.path.join(self.path, names)
        img=Image.open(img_name)
        # img = img.convert("RGB")
        # label
        if self.path_label is not None:
            return transform(img), torch.tensor(self.y[item])
        else:
            return transform(img)

    def __len__(self):
        return len(self.path_list)

def im_convert(tensor):
    """展示"""
    image=tensor.to("cpu").clone().detach()
    image=image.numpy().squeeze()
```

```
    image=image.transpose(1,2,0)
    # image=image.clip(0,1)   # numpy.clip(a, a_min, a_max, out=None) 将数组中的元素限制在a_min, a_max之间

    return image


train_path='../data/image_message'
valid_path='../data/image_message_val'
train_label='../data/train.csv'
valid_label='../data/val.csv'
batch_size=50

train_data=Mydata(train_path,train_label)
valid_data=Mydata(valid_path,valid_label)


# print(train_data[0])
# print(train_data[1])
# print(type(train_data[0]))
# print(train_data[0][0].size())

train_loader=DataLoader(dataset=train_data,batch_size=batch_size,shuffle=True,drop_last=True)
valid_loader=DataLoader(dataset=valid_data,batch_size=batch_size,shuffle=True,drop_last=True)


fig=plt.figure(figsize=(20,12))
columns=4
rows=2
dataiter=iter(train_loader)
inputs, classes=dataiter.next()

for idx in range(columns*rows):
    ax=fig.add_subplot(rows,columns,idx+1)
    n = classes[idx]   #  n:tensor(62)
    ax.set_title(n)
    plt.imshow(im_convert(inputs[idx]))
plt.show()

data_loaders={'train':train_loader,'valid':valid_loader}
```

```
model_name='resnet'
feature_extract=True   # 是否用别人训练好的
train_on_gpu=torch.cuda.is_available()
if not train_on_gpu:
    print("GPU is not available, train on CPU")
else:
    print("GPU is available, train on GPU")

device=torch.device("cuda:0" if torch.cuda.is_available() else "cpu")

def set_parameter_requires_grad(model,feature_extracting):
    if feature_extracting:
        for param in model.parameters():
            param.requires_grad=False

# model_ft=models.resnet152()
# print(model_ft)

def
initialize_model(model_name,num_classes,feature_extract,use_pretrained=True)
:
    if model_name=='resnet':
        model_ft=models.resnet152(pretrained=use_pretrained)
        set_parameter_requires_grad(model_ft, feature_extract)   # 是否冻住前面
        num_ftrs = model_ft.fc.in_features
        model_ft.fc = nn.Sequential(nn.Linear(num_ftrs, num_classes),
                                    nn.LogSoftmax(dim=1))

    return model_ft

model_ft=initialize_model('resnet',62,feature_extract)  # 几分类
model_ft=model_ft.to(device)
filename='checkpoint.pth'
para_learn=model_ft.parameters()
if feature_extract:
    para_learn=[]
    for name,parameter in model_ft.named_parameters():
        if parameter.requires_grad==True:
            para_learn.append(parameter)
            print('\t', name)
else:
    for name,pare in model_ft.named_parameters():
```

```
        if pare.requires_grad==True:
            print('\t',name)

optimizer_ft=optim.Adam(para_learn,lr=1e-2)
scheduler=optim.lr_scheduler.StepLR(optimizer_ft,step_size=7,gamma=0.1)
criterion=nn.NLLLoss()

def
train_model(models,dataloader,criterion,optimizer,num_epochs=25,filename=fi
lename):
    start=time.time()
    best_acc=0
    best_model_wts=copy.deepcopy(models.state_dict())
    train_loss=[]
    valid_loss=[]
    train_acc=[]
    valid_acc=[]
    models=models.to(device)
    LRs = [optimizer.param_groups[0]['lr']]

    for epoch in range(num_epochs):
        print('{}/{}'.format(epoch+1,num_epochs))
        for phase in ['train','valid']:
            if phase =='train':
                models.train()
                print('this is train process')
            else:
                models.eval()
                print('this is valid process')
            running_loss = 0
            running_acc = 0

            for inputs, label in dataloader[phase]:
                inputs = inputs.to(device)
                # [8, 3, 240, 108])
                label =label.to(device)
                outputs=models(inputs)
                optimizer.zero_grad()
                loss=criterion(outputs,label)
                _,pred=torch.max(outputs,1)
                if phase =='train':
                    loss.backward()
                    optimizer.step()
```

```
                running_loss+=loss.item()
                running_acc+=torch.sum(pred==label.data)
            epoch_loss=running_loss/num_epochs
            epoch_acc=running_acc.double()/len(dataloader[phase].dataset)
            time_cost=time.time()-start
            print('Time cost ',time_cost//60,'min',time_cost % 60,'s')
            print('{} loss:{:.4f}
Acc:{:.4f}'.format(phase,epoch_loss,epoch_acc))
# this

            if phase == 'valid' and epoch_acc > best_acc:
                best_acc = epoch_acc
                best_model_wts = copy.deepcopy(models.state_dict())
                state = {
                    'state_dict': models.state_dict(),
                    'best_acc': best_acc,
                    'optimizer': optimizer.state_dict(),
                }
                torch.save(state, filename)
            if phase == 'valid':
                valid_acc.append(epoch_acc)
                valid_loss.append(epoch_loss)
                scheduler.step()
            if phase == 'train':
                train_acc.append(epoch_acc)
                train_loss.append(epoch_loss)

        print('Optimizer learning rate :
{:.7f}'.format(optimizer.param_groups[0]['lr']))
        LRs.append(optimizer.param_groups[0]['lr'])
        print()


    time_elapsed = time.time() - start
    print('Training complete in {:.0f}m {:.0f}s'.format(time_elapsed // 60,
time_elapsed % 60))
    print('Best val Acc: {:4f}'.format(best_acc))

    # 训练完后用最好的一次当做模型最终的结果
    models.load_state_dict(best_model_wts)
    return models, valid_acc, train_acc, valid_loss, train_loss, LRs
```

```
# model_ft, val_acc_history, train_acc_history, valid_losses, train_losses, LRs
= train_model(
#     model_ft, data_loaders, criterion, optimizer_ft, num_epochs=3)

"""
加载模型
"""
checkpoint = torch.load(filename)
best_acc = checkpoint['best_acc']
model_ft.load_state_dict(checkpoint['state_dict'])
"""
预测数据
"""
pred=[]
test_path='../data/test_images'
test_label=None
test_data=Mydata(test_path,test_label)
train_loader=DataLoader(dataset=test_data,batch_size=batch_size,drop_last=F
alse)

for dataiter in iter(train_loader):
    images = dataiter

    model_ft.eval()

    if train_on_gpu:
        output = model_ft(images.cuda())
    else:
        output = model_ft(images)

    _, preds_tensor = torch.max(output, 1)

    preds = np.squeeze(preds_tensor.numpy()) if not train_on_gpu else
np.squeeze(preds_tensor.cpu().numpy())
    pred.extend(preds)


test_names=os.listdir('../data/test_images')
test_label=pd.DataFrame(test_names)
test_label['1']=pred
```

```
test_label.to_csv('submit.csv',index=None,header=None)
# 得到一个batch的测试数据
```

split.py

```
# -*- coding:utf-8 -*-
import os
import random
import shutil

val_percent=0.1
source_path_mess="../data/train/image_message"
source_path_origin="../data/train/image_original"

des_path_mess="../data/valid/image_message_val"
des_path_origin="../data/valid/image_original_val"

# if not os.path.exists(des_path_mess):
#     os.makedirs(des_path_mess)
#     print("创建{}文件夹".format(des_path_mess))
# if not os.path.exists(des_path_origin):
#     os.makedirs(des_path_origin)
#     print("创建"+des_path_origin+"文件夹")


def split_data(sour_path,des_path):
    if not os.path.exists(des_path):
        os.makedirs(des_path)
        print("创建{}文件夹".format(des_path))
    files1=os.listdir(sour_path)
    num1=range(len(files1))
    select_num1=int(val_percent*len(files1))
    file_select=random.sample(num1,select_num1)
    for i in file_select:
        name=files1[i]
        file_name=sour_path+'/'+name
        des_file_name=des_path+'/'+name
        shutil.move(file_name,des_file_name)


if __name__=="__main__":
    split_data(source_path_mess,des_path_mess)
    split_data(source_path_origin,des_path_origin)
```

classify.py

```
# -*- coding:utf-8 -*-
# -*- coding:utf-8 -*-

import copy
import json
import time

import pandas as pd
from torch.utils.data import Dataset,DataLoader,TensorDataset
import matplotlib.pyplot as plt
import numpy as np
import torch
import torchvision
import torch.functional as F
from PIL import Image,ImageFile
from torchvision import transforms,datasets,models
import os
from torch import nn
import torch.optim as optim
ImageFile.LOAD_TRUNCATED_IMAGES = True
print(torch.__version__)

data_dir='../data'
train_dir=data_dir+'/train'
valid_dir=data_dir+'/valid'

data_transforms={
    'train':transforms.Compose([
        transforms.RandomRotation(45),
        transforms.Resize([400, 400]),
        transforms.RandomHorizontalFlip(p=0.5),
        transforms.RandomVerticalFlip(p=0.5),

transforms.ColorJitter(brightness=0.2,contrast=0.1,saturation=0.1,hue=0.1),
        transforms.RandomGrayscale(p=0.025),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406],[0.229, 0.224, 0.225])
    ]),
```

```
    'valid':transforms.Compose([
        transforms.Resize([400, 400]),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406],[0.229, 0.224, 0.225])
    ])
}


batch_size=8
image_datasets={x:datasets.ImageFolder(os.path.join(data_dir,x),data_transf
orms[x]) for x in ['train','valid']}
data_loaders={x:torch.utils.data.DataLoader(image_datasets[x],batch_size=ba
tch_size,shuffle=True) for x in ['train','valid']}
data_size={x:len(image_datasets[x]) for x in ['train','valid']}
class_names=image_datasets['train'].classes
print(image_datasets['train'].classes)
print(image_datasets['train'].class_to_idx)

image_datasets


def im_convert(tensor):
    """展示"""
    image=tensor.to("cpu").clone().detach()
    image=image.numpy().squeeze()
    image=image.transpose(1,2,0)
    image=image*np.array((0.229, 0.224, 0.225))+np.array((0.485, 0.456, 0.406))
    image=image.clip(0,1)   # numpy.clip(a, a_min, a_max, out=None) 将数组中的
元素限制在a_min, a_max之间

    return image


fig=plt.figure(figsize=(20,12))
columns=4
rows=2
dataiter=iter(data_loaders['valid'])
inputs, classes=dataiter.next()

for idx in range(columns*rows):
    ax=fig.add_subplot(rows,columns,idx+1)
    n = classes[idx]   #  n:tensor(62)
    ax.set_title(int(classes[idx]))
```

```
    plt.imshow(im_convert(inputs[idx]))
plt.show()

model_name='resnet'
feature_extract=True   # 是否用别人训练好的
train_on_gpu=torch.cuda.is_available()
if not train_on_gpu:
    print("GPU is not available, train on CPU")
else:
    print("GPU is available, train on GPU")

device=torch.device("cuda:0" if torch.cuda.is_available() else "cpu")

def set_parameter_requires_grad(model,feature_extracting):
    if feature_extracting:
        for param in model.parameters():
            param.requires_grad=False

model_ft=models.resnet152()
model_ft
# for name,parameter in  model_ft.named_parameters():
#    print(name,parameter)
#
def
initialize_model(model_name,num_classes,feature_extract,use_pretrained=True)
:
    if model_name=='resnet':
        model_ft=models.resnet152(pretrained=use_pretrained)
        set_parameter_requires_grad(model_ft, feature_extract)   # 是否冻住前面
        num_ftrs = model_ft.fc.in_features
        model_ft.fc = nn.Sequential(nn.Linear(num_ftrs, num_classes),
                                    nn.LogSoftmax(dim=1))
        input_size = 224

    return model_ft,input_size

model_ft,input_size=initialize_model('resnet',2,feature_extract)
model_ft=model_ft.to(device)
filename='checkpoint3.pth'
para_learn=model_ft.parameters()
if feature_extract:
    para_learn=[]
    for name,parameter in model_ft.named_parameters():
```

```
        if parameter.requires_grad==True:
            para_learn.append(parameter)
            print('\t', name)
else:
    for name,pare in model_ft.named_parameters():
        if pare.requires_grad==True:
            print('\t',name)

optimizer_ft=optim.Adam(para_learn,lr=1e-2)
scheduler=optim.lr_scheduler.StepLR(optimizer_ft,step_size=7,gamma=0.1)
criterion=nn.NLLLoss()

def
train_model(models,dataloader,criterion,optimizer,num_epochs=25,filename=fi
lename):
    start=time.time()
    best_acc=0
    best_model_wts=copy.deepcopy(models.state_dict())
    train_loss=[]
    valid_loss=[]
    train_acc=[]
    valid_acc=[]
    models=models.to(device)
    LRs = [optimizer.param_groups[0]['lr']]

    for epoch in range(num_epochs):
        print('{}/{}'.format(epoch+1,num_epochs))
        for phase in ['train','valid']:
            if phase =='train':
                models.train()
            else:
                models.eval()

            running_loss = 0
            running_acc = 0

            for inputs, label in dataloader[phase]:
                inputs = inputs.to(device)
                label =label.to(device)
                outputs=models(inputs)
                optimizer.zero_grad()
                loss=criterion(outputs,label)
                _,pred=torch.max(outputs,1)
```

```
                if phase =='train':
                    loss.backward()
                    optimizer.step()


                running_loss+=loss.item()
                running_acc+=torch.sum(pred==label.data)
            epoch_loss=running_loss/num_epochs
            epoch_acc=running_acc.double()/len(dataloader[phase].dataset)
            time_cost=time.time()-start
            print('Time cost ',time_cost//60,'min',time_cost % 60,'s')
            print('{} loss:{:.4f}
Acc:{:.4f}'.format(phase,epoch_loss,epoch_acc))
# this

            if phase == 'valid' and epoch_acc > best_acc:
                best_acc = epoch_acc
                best_model_wts = copy.deepcopy(models.state_dict())
                state = {
                    'state_dict': models.state_dict(),
                    'best_acc': best_acc,
                    'optimizer': optimizer.state_dict(),
                }
                torch.save(state, filename)
            if phase == 'valid':
                valid_acc.append(epoch_acc)
                valid_loss.append(epoch_loss)
                scheduler.step()
            if phase == 'train':
                train_acc.append(epoch_acc)
                train_loss.append(epoch_loss)

        print('Optimizer learning rate :
{:.7f}'.format(optimizer.param_groups[0]['lr']))
        LRs.append(optimizer.param_groups[0]['lr'])
        print()


    time_elapsed = time.time() - start
    print('Training complete in {:.0f}m {:.0f}s'.format(time_elapsed // 60,
time_elapsed % 60))
    print('Best val Acc: {:4f}'.format(best_acc))
```

```
    # 训练完后用最好的一次当做模型最终的结果
    models.load_state_dict(best_model_wts)
    return models, valid_acc, train_acc, valid_loss, train_loss, LRs

# model_ft, val_acc_history, train_acc_history, valid_losses, train_losses, LRs
= train_model(
#     model_ft, data_loaders, criterion, optimizer_ft, num_epochs=3)


# 加载模型
checkpoint = torch.load(filename)
best_acc = checkpoint['best_acc']
model_ft.load_state_dict(checkpoint['state_dict'])
"""
验证集
"""
transform=transforms.Compose([transforms.ToTensor(),
                          transforms.Resize((400,400)),
                          transforms.Normalize([0.485, 0.456, 0.406],[0.229,
0.224, 0.225])
                          ])

class Mydata3(Dataset):
    def __init__(self,path):
        self.path=path
        self.file_list=os.listdir(self.path)
    def __getitem__(self, item):
        names = self.file_list[item]
        img_name = os.path.join(self.path, names)
        img = Image.open(img_name)
        return transform(img)
    def __len__(self):
        return len(self.file_list)


test_data=Mydata3("../data/test_images")
test_loader=DataLoader(dataset=test_data,batch_size=batch_size,drop_last=Fa
lse)
pred=[]
for dataiter in iter(test_loader):
    images = dataiter

    model_ft.eval()
```

```
    if train_on_gpu:
        output = model_ft(images.cuda())
    else:
        output = model_ft(images)

    _, preds_tensor = torch.max(output, 1)

    preds = np.squeeze(preds_tensor.numpy()) if not train_on_gpu else
np.squeeze(preds_tensor.cpu().numpy())
    pred.extend(preds)


test_names=os.listdir('../data/test_images')
test_label=pd.DataFrame(test_names)
test_label['1']=pred

test_label.to_csv('submit3.csv',index=None,header=None)


# def process_image(image_path):
#     # 读取测试数据
#     img = Image.open(image_path)
#     # 相同的预处理方法
#     mean = np.array([0.485, 0.456, 0.406])  # provided mean
#     std = np.array([0.229, 0.224, 0.225])  # provided std
#     img = (img - mean) / std
#
#     # 注意颜色通道应该放在第一个位置
#     img = img.transpose((2, 0, 1))
#     return img
#
#
# def imshow(image, ax=None, title=None):
#
#     if ax is None:
#         fig, ax = plt.subplots()
#
#     # 颜色通道还原
#     image = np.array(image).transpose((1, 2, 0))
#
#     # 预处理还原
#     mean = np.array([0.485, 0.456, 0.406])
```

```
#     std = np.array([0.229, 0.224, 0.225])
#     image = std * image + mean
#     image = np.clip(image, 0, 1)
#
#     ax.imshow(image)
#     ax.set_title(title)
#
#     return ax
#
# image_path = '../data/test_images/im19951.jpg'
# img = process_image(image_path)
# imshow(img)

# 得到一个batch的测试数据
dataiter = iter(data_loaders['valid'])
images, labels = dataiter.next()

model_ft.eval()

if train_on_gpu:
    output = model_ft(images.cuda())
else:
    output = model_ft(images)

_, preds_tensor = torch.max(output, 1)

preds = np.squeeze(preds_tensor.numpy()) if not train_on_gpu else
np.squeeze(preds_tensor.cpu().numpy())
preds


fig=plt.figure(figsize=(20, 20))
columns =4
rows = 2

for idx in range (columns*rows):
    ax = fig.add_subplot(rows, columns, idx+1, xticks=[], yticks=[])
    plt.imshow(im_convert(images[idx]))
    ax.set_title("{} ({})".format(str(preds[idx]), str(labels[idx].item())),
                 color=("green" if str(preds[idx])==str(labels[idx].item()) else
"red"))
plt.show()
```

result.py

```
import pandas as pd
import numpy as np

df_2=pd.read_csv("../../code/submit.csv",header=None,encoding='utf-8')   #
任务二模型的预测结果
df_3=pd.read_csv("./submit3.csv",header=None,encoding='utf-8')          # 识
别有无水印模型预测结果


# print(df_2.head(3).append(df_2.tail(3)))
# print('-'*8)
# print(df_3.head(3).append(df_3.tail(3)))
image={'图像编号':list(df_2.iloc[:,0])}
data=pd.DataFrame(image)
# print(data)
data['嵌入的信息']='无'
for i in range(len(data)):
    if df_3.iloc[i,1]==0:
        num=df_2.iloc[i,1]
        if num>=36 and num<=61:
            num=num+61
            data.iloc[i,1]=chr(num)
        elif num>=10 and num <=35:
            num=num+55
            data.iloc[i, 1] = chr(num)
        elif num>=0 and num <=9:
            num=num+48
            data.iloc[i, 1] = chr(num)

data.to_csv('./result.csv',index=None,encoding='utf_8_sig')



print(data)
data.info()
```