同济大学

毕业设计(论文)开题报告

（适用于工科类、理科类专业）

课题名称　毕业设计(论文)(通信工程)

副 标 题　低质量 QR 码的图像增强及快速识别

学　院　电子信息工程学院

专　业　通信工程

学生姓名　斯提凡　学 号　1656038

2020　年　2　月 20　日

概述：

1. 引言
2. 目标
3. 假说
4. 材料和方法
5. 预期成绩
6. 时间表
7. 资料来源

1.1 引言

随着智能手机的日益普及，QR 码（二维码）已成为一种用于获取日常生活中特定对象或事件信息的流行形式，并且许多手机应用程序都具有扫描 QR 码的功能。 QR 码的基本框架旨在嵌入具有容错功能的机器可读消息，并且生成结果通常看起来像是随机的 2D 矩阵。由于 QR 码是使用 Reed-Solomon（RS）纠错码编码的，因此 QR 扫描仪在解码内容时不必正确看到每个像素。这种纠错机制可以包容一些错误，并使 QR 码设计者可以在一定程度上更改 QR 码的外观。在过去的几年中，设计人员试图通过添加美学元素或重新设计来美化 QR 代码 ，而仍然保持信息可被识别。通常，设计人员针对两种变体进行操作以使原始 QR 码在视觉上令人愉悦，即更改模块的形状或颜色并将图片嵌入 QR 码中

快速扫描是 QR 码的另一个特点。我们来看看通常是如何使用 QR 码响应来嵌入消息，以便人们可以方便地使用移动设备识别 QR 码并由此获取信息的：我们首先将信息嵌入到 QR 码中，而后无论我们要如何读取该信息，都无需发出服务器请求。 这意味着如果通过扫描二维码访问网站，扫描二维码这个步骤无需等待服务器响应，因此很快就能打开这个网站。

此外，我还将介绍一种用于使嵌入图像变形的错误感知扭曲技术，以使生成的 QR 码中的错误最小化，并优化代码的可读性。他们通过使用二进制示例来重塑模块的规则正方形，以使其局部外观类似于示例形状。当嵌入的图像较小时，Lin 的方法适用，而不幸的是，当我们希望嵌入的视觉内容占据相当大的比例时，方法便不再适用。。

此操作的主要挑战之一是 QR 码出现在摄像头的时间必须足够长，以使传感器拍摄出高质量的图像并从中解码信息。正在移动的 QR 码可能显得模糊，导致解码无法成功。而这也违背了二维码快速扫描的特点

2.2 目标

◆ QR 码的结构

QR 码由排列在白色背景上的正方形网格中的黑色模块组成。可以由成像设备读取。 QR 码的基本信息单位是模块，是最小平方元素。QR 码中存储的信息量还与 QR 码的版本有关。带有流程图的主要建议是使用建议的带有几个模块的系统设计的。 系统的每个模块都已进行了详细描述，这似乎要求我们通过内置在手机中的摄像头捕获图像

训练时运用了逐步提升分辨率的 CNN 训练方法。为此生成了对应的高分辨率和低分辨率的 QR 码图像的数据集，来训练我们的 CNN 网络。首先算法使用低分辨率 QR 码，其尺寸相对较小，用于验证算法是否按预期工作。它们是测试和调试代码的良好起点，有利于算法的快速迭代。然后切换到高分辨率的 QR 码训练，使得算法达到最佳精准度。

在第一稿中，我训练了 CNN，其中使用了 100 张图像来训练网络，使用 50 张图像来评估网络学习对图像进行分类的准确程度。我可以直接从 TensorFlow 访问 QR 码数据集。并且直接从 TensorFlow 导入和加载数据。

预处理数据

在训练网络之前，将对数据进行预处理。 我会检查训练集中的第一张图像，我将能够看到像素值落在图像范围内, 它还将验证数据格式是否正确。并且已经构建好网络结构和计算图，将从训练集中加载第一张图像并显示预测结果。

现在，研究目标是构建快速识别系统以增强 QR 码图像的质量，从而提高解码成功率、减少读取 2D 条形码中包含的数据所需的时间。 该系统是具有自编码器结构的卷积神经网络（auto-encoder CNN），使用 TensorFlow 深度学习框架对自生成图像进行训练。 输入是分辨率和质量较差的图像，输出是相同 QR 码的清晰、无噪点的图像。 然后，CNN 的输出可用于通过任何常见的解码算法对 QR 码的数据进行解码。 该系统已部署在 Raspberry Pi 3B +上，并在不同的仿真中进行了评估。实验在收集的数据集上解码低质量 QR 码的成功率达到了约 100%。

3.3 假说

4.4 材料和方法

- ✓ 图像去模糊和增强
- ✓ CNN（Tensorflow, Pytorch）模型
- ✓ CNN 模型训练
- ✓ 硬件平台

✓ 图像去模糊和增强

图像增强系统的目的是拍摄低质量的图像，对其进行处理以获得更具可读性的图像，最后将其输入解码器。 图像质量低 来自不同的原因：

- 低图像分辨率。
- 光线条件不理想。
- 失真。
- 噪声
- 运动模糊

此要求很重要，因为如果不单独设计，图像处理算法可能需要大量时间

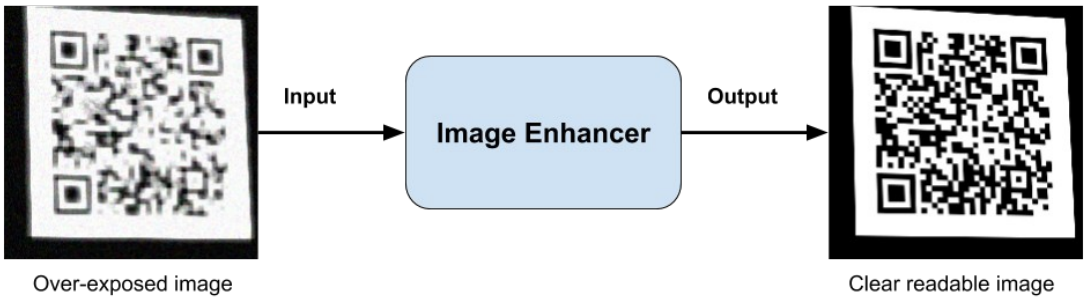


图 1 增强器输入输出

✓ CNN（Tensorflow, Pytorch）模型

开发的增强器基于卷积神经网络或简称为 CNN，从零开始经过训练，并建立在通常称为自动编码器的结构上。 CNN 是神经网络（NN），用于模拟卷积运算。卷积 I * K 是这样进行的：输入矩阵 I 和另一个矩阵（内核或滤波器 K），通过对应像素相乘得到的 I 上的内核，然后进行求和，得到对应像素的值，如图 7 所示。输出形状取决于 I，K 和其他一些参数，如矩阵的形状以及新矩阵填充（padding）大小和跨度（stride）。

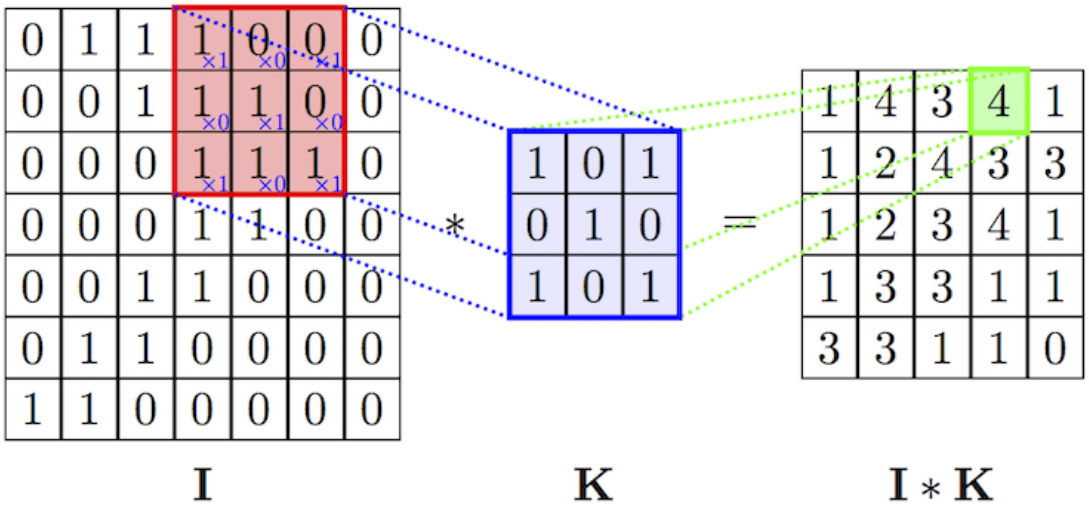


图 2　输入图像（矩阵）I 和滤波器 K 之间的卷积

在本文设计的 CNN 中，卷积执行了四次。第一次从单个原始图像（未应用填充），带有 32 个不同的 3*3 滤镜。这个运算的输出是一组 32 张图像，每张图像均来自原始图像与不同的过滤器，尺寸略有减小。然后将这些矩阵的大小减小到 XXXX。在图 2 中，通过仅保留组成图像的 2*2 的最大值。此操作是所谓的最大池化操作。然后，这些图像通过 16 个不同的内核（3*3）执行第二轮卷积。并将得到的结果图像通过第二个最大池化操作。

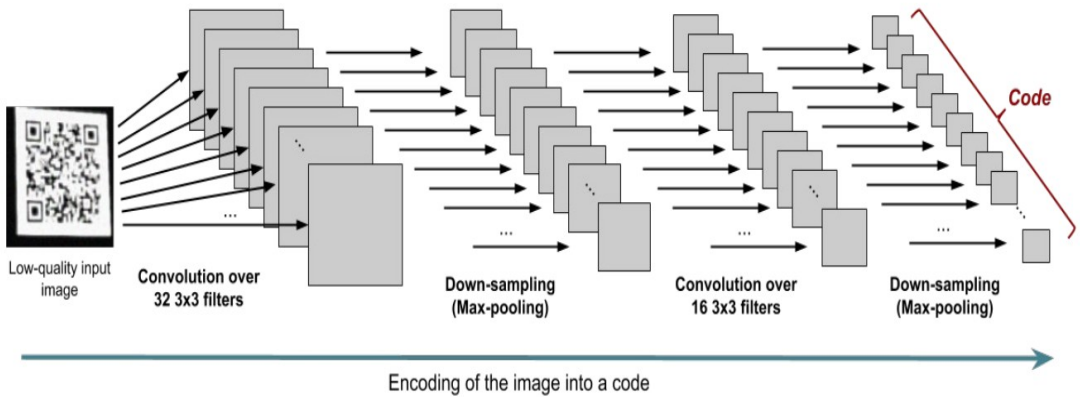


图 3 CNN 第一部分的图片

此时，图像的尺寸比原始图像小得多。通过对原始图像的操作已经获得了代表自动编码器中的代码结构体的图像。此隐含编码用于重新计算非模糊图像。在第三个矩阵中输入矩阵用 16 个滤波器（3*3）进行卷积，通过应用此时间填充，使图像获得更大的尺寸。在上采样操作后，图像的尺寸增加了一倍。再进行最后的卷积和执行上采样，以获得单个输出图像。

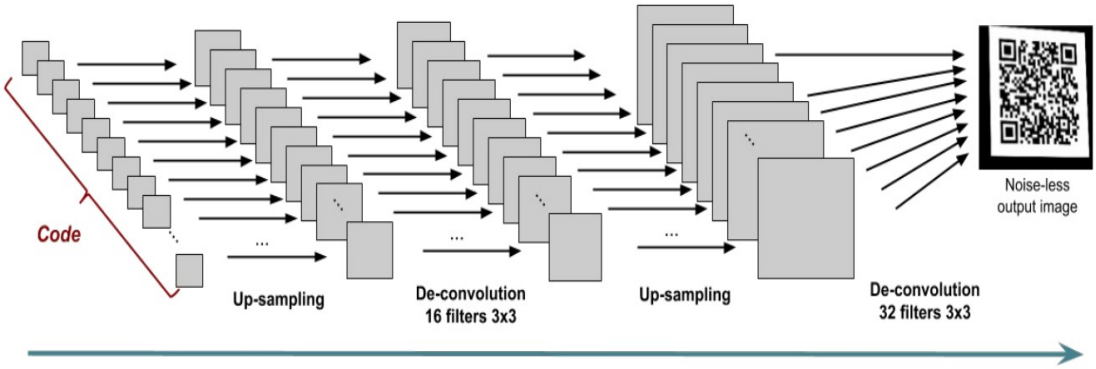


图 4 CNN 第二部分的表示。

✓ CNN 模型训练

CNN 模型包含大小为 3*3 的 96 个不同的过滤器，这意味着它包含 864 个独立参数。所有这些值一一选择是不合理的。相反，每个参数都是通过数据学习的。训练数据由许多样本对组成，由输入图像 x 及预测目标 y。由于训练了 CNN 的参数需要大量样本，首先必须生产真实的数据。

为达到此目的，项目使用了一个开源 python 库。首先输入包含随机的 10,500 个 QR 码、长度在 50 到 150 个字符之间的数据。然后将每个图像调整为 200*200 的形状，并使用 OpenCV 库进行变形，该变形也已进行随机设置参数，以确保没有图像具有完全相等的变换。这一步是必要的，以得到在真实情况下模拟图像的变形。 第三步是增加运动：在随机方向上模糊，同时修改图像的曝光，模拟 在不同光线环境下的图像。最后，添加了随机选择的校验噪声的所有图像。

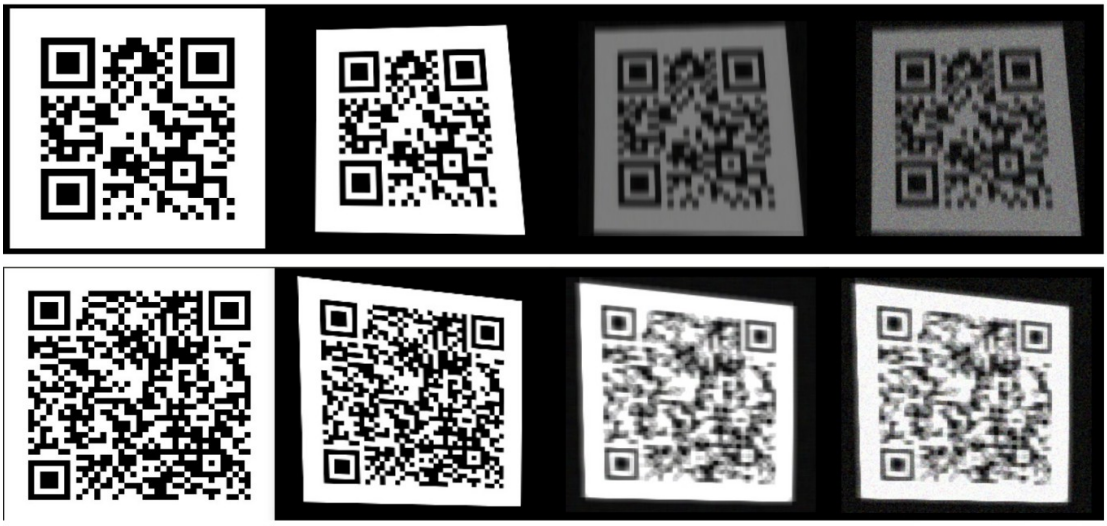
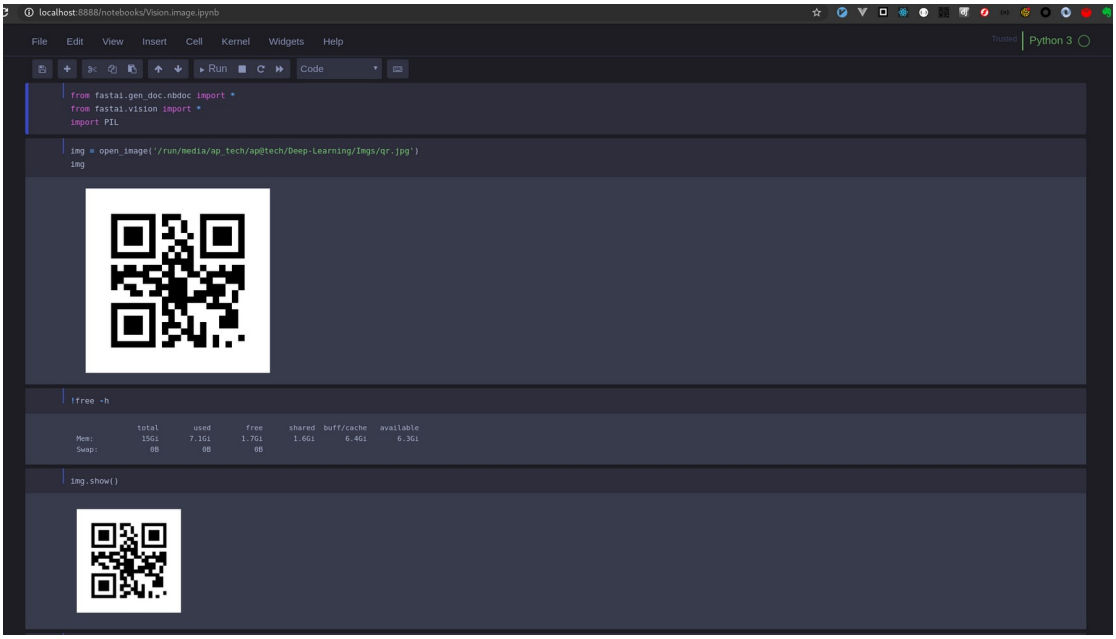


图 4 创建培训数据的过程显示为两个示例。

✓ 硬件平台

除了 Python 和下属框架外，也将使用 jupyter-notebook 编写我们的代码并测试我们的 CNN 网络



训练 CNN

```
Train on 60000 samples
Epoch 1/3
60000/60000 [==============================] - 6s 103us/sample - loss: 0.2664 - accuracy: 0.9221
Epoch 2/3
60000/60000 [==============================] - 4s 72us/sample - loss: 0.1204 - accuracy: 0.9634
Epoch 3/3
60000/60000 [==============================] - 5s 81us/sample - loss: 0.0881 - accuracy: 0.9743


   <tensorflow.python.keras.callbacks.History at 0x7fa4cdd72610>
```

CNN 的预测

```
[[1.9655302e-12 5.3892290e-12 2.9700695e-07 ... 9.9999833e-01
  2.4433058e-09 5.2575118e-09]
 [1.1762396e-11 1.1093241e-05 9.9998796e-01 ... 7.7411029e-12
  8.5119084e-10 2.2198654e-15]
 [2.7633408e-11 9.9993980e-01 1.8787377e-07 ... 1.3141019e-05
  1.9665864e-05 1.9468044e-07]
 ...
 [1.7531111e-12 4.1644773e-07 1.2153874e-08 ... 4.6645787e-06
  6.8748654e-06 2.1475547e-05]
 [9.6005888e-09 2.5228557e-08 5.1111876e-10 ... 1.8661863e-08
  2.1467071e-05 1.1884702e-10]
 [1.6842124e-10 1.3422961e-12 3.0880819e-11 ... 7.6660367e-16
  2.6003264e-11 4.7436047e-14]]
```

```
for ax in axs.flatten():
    img,bbox = get_bb_ex()
    img = img.apply_tfms(tfms[0], size=224)
    bbox = bbox.apply_tfms(tfms[0], do_resolve=False, size=224)
    img.show(ax=ax, y=bbox)
```

Embed the QR code into the image

```
import qrcode
from PIL import Image

img_bg = Image.open('/run/media/ap_tech/ap@tech/Deep-Learning/Imgs/Ap-wex.jpg')

qr = qrcode.QRCode(box_size=2)
qr.add_data('/run/media/ap_tech/ap@tech/Deep-Learning/Imgs/Ap-wex.jpg')
qr.make()
img_qr = qr.make_image()

pos = (img_bg.size[0] - img_qr.size[0], img_bg.size[1] - img_qr.size[1])

img_bg.paste(img_qr, pos)
img_bg.save('/run/media/ap_tech/ap@tech/Deep-Learning/data/save.png')
```

有关此研究的所有信息都将上传到我的个人 GitHub 帐户中：

https://github.com/AurioPinto/Thesis

6.6 论文大纲

1. 引言
    a. 动机
    b. 目标与假设
2. 背景
    a. 影像增强，图像去模糊和增强
    b. CNN 模型
    c. 二维码解码
3. 材料
    a. CNN（Tensorflow, Pytorch）模型
    b. fastai
    c. Google
        i. 为 CNN 创建数据
    d. 训练数据检索
    e. 培训过程
    f. 结果
4. 方法
    a. 文学研究
    b. 底物分析
    c. 系统评估
        i. 评估方案
        ii. 硬件平台
        iii. 结果
5. 预期结果
    a. 文献结果
    b. 基板参数
    c. 孵化结果
        i. 产生的 QR 码生产率
        ii. 不同 CNN 的比较
            - 失去 CNN
            - 低质量图像的发酵残留物
            - 系统评估
6. 讨论区
    a. QR 码产生率
    b. 错误分析
    c. QR Code 图片增加成功率解码算法
7. 结论
    a. 假设回答了吗?
    b. 进一步研究需求

7.7 资料来源

[1.] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S.rving, G., Isard, M. and Kudlur, M., 2016, November. T ensorflow: a system for large-scalemachine learning. In OSDI (Vol. 16, pp. 265-283).

[2.] Hradiš, M., Kotera, J., Zemcík, P. and Šroubek, F., 2015. Convolutional neural networks for direct text deblurring. In Proceedings of BMVC (Vol. 10, p. 2).

[3.] LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P., 1998. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), pp.2278-2324.

[4.] Liu, N., Zheng, X., Sun, H. and Tan, X., 2013. Two-dimensional bar code out-of-focus deblurring via the increment constrained least squares filter. Pattern Recognition Letters, 34(2), pp.124-130.

[5.] Munoz-Mejias, D., Gonzalez-Diaz, I. and Diaz-de-Maria, F., 2011. A low-complexity preprocessing system for restoring low-quality QR code images. IEEE Transactions on Consumer Electronics, 57(3).

[6.] Svoboda, P., Hradiš, M., Maršík, L. and Zemcík, P., 2016, September. CNN for license plate motion deblurring. In Image Processing (ICIP), 2016 IEEE International Conference on (pp.3832-3836). IEEE.

[7.] Van Gennip, Y., Athavale, P., Gilles, J. and Choksi, R., 2015. A regularization approach to blind deblurring and denoising of qr barcodes. IEEE Transactions on Image Processing, 24(9), pp.2864-2873.

[8.] Xu, W. and McCloskey, S., 2011, January. 2D Barcode localization and motion deblurring using a flutter shutter camera. In Applications of Computer Vision (WACV), 2011 IEEE Workshop on (pp. 159-165). IEEE.

[9.] Zbar Website. URL: http://zbar.sourceforge.net/. Accessed in December 2018.

[10.] GoogleCharts.URL:https://developers.google.com/chart/infographics/docs/ qr_codes

[11.] Training a classifier URL: https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html

[12] CNN with pytorchURL: