****

本科毕业设计（论文）

**嵌入式人工智能实验系统平台设计**

|  |  |
| --- | --- |
| **学院**  **专业**  **学生姓名**  **学生学号**  **指导教师**  **提交日期** | **电子与信息学院** |
| **电子科学与技术** |
| **成泽森** |
| **201730271412** |
| **高学** |
| **年 月 日** |

# 摘 要

近年来, 由于深度学习的崛起, 让人工智能领域开始高速发展, 尤其是深度学习在计算机视觉上的应用, 这为, 例如, 医疗, 遥感, 无人车等等领域, 提供了极大的支持。 我们期望能够搭建一个深度学习实验系统, 来对当前火热的深度学习技术做一个实验演示, 能够帮助他人更好地了解深度学习技术。为使得深度学习实验系统更便携, 更轻量, 我们使用一些计算资源比较弱的边缘设备作为系统的主要计算设备。

实现这个深度学习实验系统主要分为三步: 首先我们需要在强计算资源平台 (例如,小型台式电脑, 小型笔记本, 集群) 复现各种经典的网络, 完成功能的实现; 而后我们将网络移植部署到弱计算资源平台 (例如, Jetson Nano, Raspberry Pi), 完成相关功能的演示; 而后我们会尝试一些模型压缩的方法, 例如, 设计和改造网络架构, 知识蒸馏等, 来提升整个演示的效果。

具体来说, 第一步, 我们主要实现两个计算机视觉任务: 人脸检测与人手检测。 首先我们会尝试一些通用的目标检测网络, 例如, Yolov3 [1] 等, 来做人脸检测和人手检测这两个任务, 而后我们会使用 RetinaFace [2] 这一专用的人脸检测网络来做人脸检测任务, Hand-CNN [3] 这一专用的人手检测网络来做人手检测。第二步, 我们会尝试部署部分网络架构到弱计算资源平台上, 并接入一些外围电路来进行展示。 其中有一些网络由于体量过大, 就算模型压缩之后也难以移植到弱计算资源平台, 所以我们只选择部分网络来进行部署。第三步, 我们会对移植到弱计算资源平台的网络进行一些模型压缩相关的操作, 来使其更加轻量级, 从而在若计算资源平台上有着更好地表现。

**关键词**：边缘设备; 目标检测; 模型压缩; 模型部署;

# Abstract

In recent years, artificial intelligence becomes really popular because of the development of deep learning technique especially deep learning based computer vision technique. Deep learning based computer vision technique boosts remote sense, medical diagnosis, autonomous car, etc. We expect to build a deep learning experiment system to make some demos about deep learning based computer vision technique, which may let people to have a better understand of this technique. In order to make system more lightweight, we use edge device with poor compute resources as the main compute device of system.

There are three steps for implementation of this experiment system: At first, we will implement some classical deep learning models on strong compute sources platform (for example, desktop, laptop, cluster); Then, we will try to deploy these models to weak compute sources platform (for example, Jetson nano, Raspberry Pi); At last, we will try to compress the models to let them more efficient. We may use architecture modification and knowledge distillation to conduct model compression.

In detail, we may focus on hand detection and face detection firstly. On one hand, we will implement dedicated detection models like RetinaFace [2] and Hand-CNN [3] to perform hand detection and face detection, respectively. On the other hand, we will implement general detection models like YOLOv3 [1] to perform both hand detection and face detection. All of the implementations will be conducted on strong compute sources platform. Secondly, we deploy those models mentioned above to weak compute sources platform. Then, we will integrate the compute platform with functional circuits to demo our implemented models. Some of the models will be ignored because of their large parameters when deploying to weak compute sources platform. Thirdly, we will conduct model compression on selected models, which will make them more lightweight and more efficient.

**Keywords**: Edge Device; Object Detection; Model Compression; Model Deployment;

# 目 录

[摘 要 II](#_Toc71681773)

[Abstract III](#_Toc71681774)

[目 录 IV](#_Toc71681775)

[第一章 绪论 1](#_Toc71681776)

[1.1 引言 1](#_Toc71681777)

[1.2 研究背景 1](#_Toc71681778)

[1.3 研究现状 1](#_Toc71681779)

[1.4 论文结构 2](#_Toc71681780)

[第二章 卷积神经网络 3](#_Toc71681781)

[2.1 卷积 3](#_Toc71681782)

[2.1.1 前向传播 3](#_Toc71681783)

[2.1.2 反向传播 5](#_Toc71681784)

[2.2 池化 6](#_Toc71681785)

[2.2.1 前向传播 6](#_Toc71681786)

[2.2.2 反向传播 8](#_Toc71681787)

[2.3 激活函数 9](#_Toc71681788)

[2.4 全连接层 10](#_Toc71681789)

[2.4.1 前向传播 11](#_Toc71681790)

[2.4.2 反向传播 11](#_Toc71681791)

[2.5 批正则化 12](#_Toc71681792)

[2.6 特征提取 13](#_Toc71681793)

[第三章 目标检测 16](#_Toc71681794)

[3.1 任务定义 16](#_Toc71681795)

[3.2 one-stage 检测器 18](#_Toc71681796)

[3.3 two-stage 检测器 18](#_Toc71681797)

[3.4 anchor-free 检测器 18](#_Toc71681798)

[第四章 手写数字及写字人识别实验过程及其结果 19](#_Toc71681799)

[4.1 手写数字识别实验 19](#_Toc71681800)

[4.1.1 样本简介 19](#_Toc71681801)

[4.1.2 Writer Depend类数字识别实验 19](#_Toc71681802)

[4.1.3 Writer Depend类数字识别实验结果分析 20](#_Toc71681803)

[4.1.4 Writer Independ类数字识别实验 21](#_Toc71681804)

[4.1.5 样本简介 21](#_Toc71681805)

[4.1.6 两位写字人识别实验 22](#_Toc71681806)

[4.2 本章小结 22](#_Toc71681807)

[结论 23](#_Toc71681808)

[1. 论文工作总结 23](#_Toc71681809)

[2. 工作展望 23](#_Toc71681810)

[参考文献 24](#_Toc71681811)

[致谢 28](#_Toc71681812)

# 绪论

## 引言

近年来, 由于科技的不断进步, 计算资源的不断丰富, 设备计算能力的不断增强, 计算密集型的人工智能技术开始兴起, 例如, 广为人知的 AlphaGo, 腾讯绝悟 AI, 谷歌无人车等等, 当然目前最受人瞩目, 最被人期待, 影响力最高, 发展最为完善的当属基于深度学习的计算机视觉技术, 深度学习为计算机视觉技术带来了革新, 使得在过去一些比较艰深困难的任务变得可以解决, 例如, 人脸识别技术, 现在已经在性能上超越人类, 同时也引入了一大批新的任务, 例如, GAN 带来的 AI 换脸, 这在此前都是难以想象的。

## 研究背景

基于深度学习的计算机视觉技术受人瞩目, 而且其早已融入到我们生活的方方面面,例如, 高铁站的人脸安检系统, 单位的车牌识别系统等等, 在此背景下, 有越来越多各行各业的人开始学习基于深度学习的计算机视觉技术, 然而想要真正了解基于深度学习的计算机视觉技术, 而没有其他助力的情况下, 这个过程是相对有点长的, 我们希望通过构建一个实验系统来为深度学习的初学者提供一些演示和指南, 在实验系统上实践的同时, 加深对于基于深度学习的计算机视觉技术的理解和认知。与此同时, 我们还会开源一份 codebase, 以便大家修改, 在理解和认知基于深度学习的计算机视觉技术上能够快速上手, 着手于搭建自己的深度学习模型。

## 研究现状

真正来说, 搭建一个关于基于深度学习的计算机视觉技术的实验平台很难算得上是一项研究, 因为在单纯地搭建实验平台的过程中, 我们没有引入新的问题, 也不会提出新的关于问题的解决方案, 考虑到实验平台演示的技术主要为目标检测, 在尝试目标检测的过程中, 我们可能会提出新的方案, 所以目标检测这一任务更带有研究性质, 那么我们主要对目标检测的研究现状做一个分析, 由于目标检测引入深度学习前后, 整个体系也有着比较大的差异, 所以我们只分析引入深度学习之后的发展过程, 目标检测网络最初主要为 two-stage 检测网络, two-stage 检测网络把目标的检出拆分成了两个部分, 第一个 stage 主要用于定位, 第二个 stage 主要用于分类, 将两个 stage 结合起来, 我们就能知道这个物体在哪里? 是什么? two-stage 的网络有很多, 例如, 大家熟知的 R-CNN 系列 (R-CNN [4], Fast R-CNN [5], Faster R-CNN [6], Mask R-CNN [7], Cascaded R-CNN [8]), 引入 ROI pooling 来使得不同大小的物体能够输出对齐的特征大小用于分类的 SPPNet [9], 引入特征金字塔来提取多尺度特征的 FPN [10] 等等; two-stage 检测网络能够初步地解决目标检测这一任务, 但是又引入了很多问题, 例如, 推理速度太慢, Faster R-CNN 在 Nvidia Tesla k40 GPU 上推理速度仅为 5 fps [6]; 需要引入锚框等先验知识, 必须得提前制定好一系列用于放缩先验框, 而且先验框的设定会很大程度上影响性能。上述两个问题, 就导致了后面两种发展方向: one-stage 检测网络和 anchor-free 检测网络。 one-stage 检测网络主要为了解决推理速度过慢的问题, 在精度上比 two-stage 网络稍差, 比较著名的 one-stage 网络有 YOLO 系列 (YOLOv1 [11], YOLOv2 [12], YOLOv3 [1]), SSD 系列 [13], RetinaNet [14] 等等; anchor-free 网络则解决了必须引入先验锚框的问题, 而且 anchor-free 往往同时是 one-stage 网络, 例如, FCOS [15], anchor-free 网络在速度和精度上的表现都比较优越, 所以 anchor-free 会是目标检测领域在今后探索的一个比较重要的方向。

## 论文结构

本文分为四章, 虽然我们最终是为了实现一个实验系统, 然而在实验系统上搭载的任务主要以目标检测为主, 而且整个搭载的过程乃是这项工作的核心, 所以整篇文章会以目标检测相关的工作来推进, 其中第一章简述了实验系统的必要性, 以及实验系统上目标检测技术的发展现状; 第二章将重点介绍目标检测技术用到的神经网络结构 – 卷积神经网络; 第三章主要拆解分析目标检测技术, 并详细分析这篇文章的工作涉及到的相关目标检测网络; 与此同时我们会将完全处理好的模型部署到弱计算资源平台上, 并制作一些演示的 demo 以供展示。

# 卷积神经网络

## 卷积

卷积神经网络是一种深度学习技术的算法, 其天然地符合计算机视觉的计算范式, 卷积神经网络对图像的处理可以看作是引入深度学习之前那些图像处理技术的继承。传统的图像处理技术, 例如, Canny 算子, Sobel 算子, Laplace 算子, 实际上与当前的卷积神经网络中的卷积算子有一定的兼容性。卷积是整个卷积神经网络的核心, 卷积层有两大优势, 平移等变性与局部计算 [16]。其中平移等变性使得当对输入卷积层的图像做一些翻转, 平移等处理的时候, 卷积层依然能完成相应的功能; 局部计算则使得单层卷积层能够提取局部特征信息, 而堆叠之后的多层卷积又能够提取全局特征信息 [17]。这两个特性使得卷积神经网络能够发挥出非常好的性能。在分析卷积神经网络的功能之前, 我们先具体分析一下卷积的细节。

### 2.1.1 前向传播

卷积有很多种, 在这里由于我们专注的领域是计算机视觉领域, 所以我们只分析针对于二维图像这一欧几里得结构的数据的二维卷积操作, 对于其他领域还有一些非常规的卷积, 例如, 图神经网络的图卷积等等。下图是二维卷积操作的演示图:

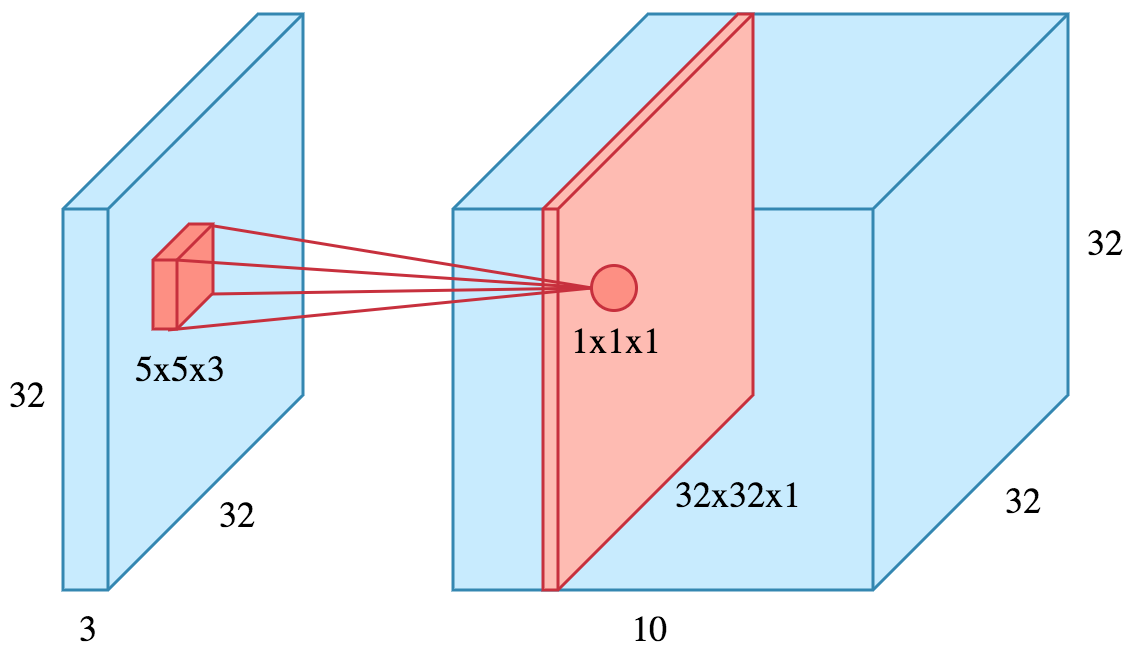


图1. 二维卷积操作示意图

输入一般为二维的 RGB 图像或者网络的中间特征图 , 一般会有 H (height), W (width), C (channel) 三个维度, 与图像或特征图发生计算的主要是卷积核 , 二维卷积核是一个四维的算子, 为卷积核的尺寸, 为卷积核为了对齐输入图像或者特征图的通道数, 而 则是为了将输入图像或者特征图形变为输出特征图的通道数, 卷积核主要在 H 和 W 的维度上局部滑动, C 通道会全部纳入计算, 假设输入特征图为 , 卷积核为 , 输出特征图为 , 则卷积的计算公式如下:

(1)

计算细节示意图:

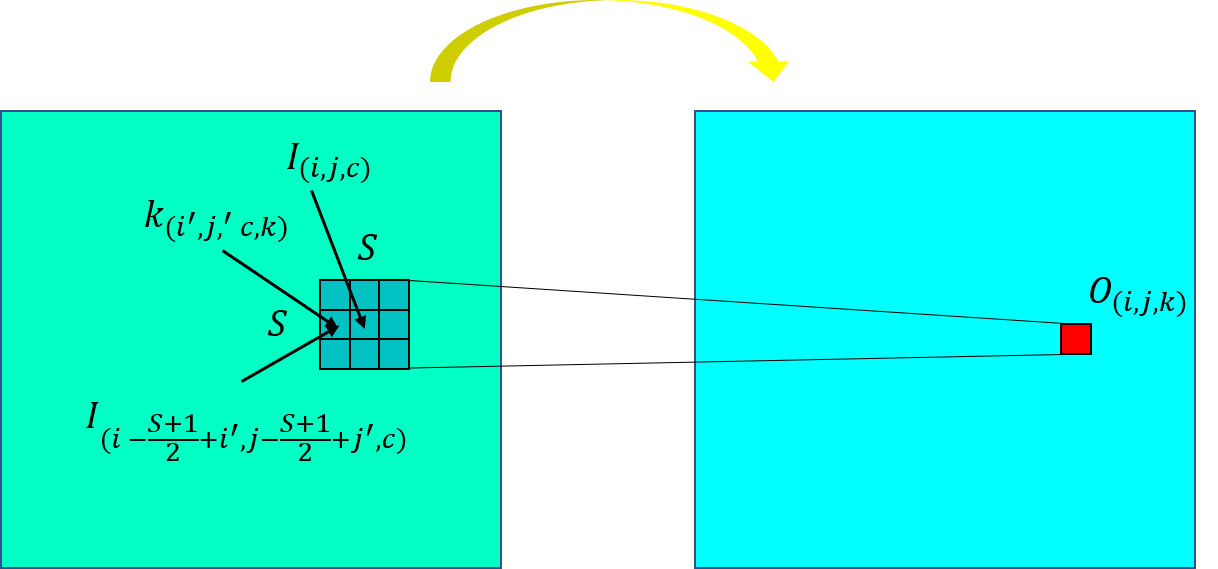


图2. 卷积计算细节图

其中 代表特征图或者图像的行, 代表特征图或者图像的列, 代表输入特征图的通道下标, 代表输出特征图的通道下标。

卷积核往往是长宽相等的, 对称的, 不过这并不绝对, 例如, ACNet 提出了非对称卷积, 通过分支融合的方式结合各种非对称卷积得到的信息 [18]。卷积核的尺寸 往往选为 3 或者 1, 3x3 和 1x1 的卷积核功能不同, 往往 3x3 的卷积核用于特征提取, 而 1x1 的卷积核用于特征降维, 1x1 的卷积核是独特的, 而 3x3 卷积却并不独特, 仔细考虑卷积的计算公式就可以得知, 实际上用 5x5, 7x7 等尺寸也能够达到相同的效果, 为什么一般只使用 3x3 这一尺寸呢？在卷积神经网络发展的早期, 非常流行使用大卷积核, 例如, AlexNet [19] 中使用了 7x7 与 5x5 的卷积层作为中间特征提取器, 而后研究者们发现 7x7, 5x5 等大奇数卷积核可以被拆分成好几个 3x3 的小奇数卷积核, 最终的计算效果一样, 为了保持卷积层的颗粒度, 所以就保持着 3x3 卷积了。

把卷积的计算过程讲解清楚了, 我们再对卷积的一些超参数进行讨论, 卷积的各种控制参数实际上不止卷积核的尺寸 (kernel size) 这一个, 在之前对于卷积的分析中, 我们都是假定卷积的滑动步长为 1, 实际上卷积的步长 (stride) 是可以控制的, 此外由于卷积会有一些边界问题会导致输入的特征图和图像的尺寸发生变化, 所以往往还有填充操作 (padding) 用于补偿这部分尺寸变化, 注意, 这里的关于尺寸的变化只针对 H, W 这两个通道, 对于 C 通道的变化取决于卷积核的 和 , 这里给出卷积核尺寸 (kernel size), 步长 (stride), 填充操作 (padding) 这些参数对尺寸的改变公式:

(2)

(3)

其中 , 表示输出特征图的长宽, , 表示输入特征图的长宽, 代表填充大小, 代表卷积核尺寸, 代表步长, “” 代表向下取整操作。

### 2.1.2 反向传播

我们已经了解了卷积的计算过程, 在这之后我们不得不说一说卷积的反向传播过程, 卷积是卷积神经网络中的构成, 而卷积神经网络是一种深度学习模型, 这个模型可以看作是一个函数, 我们通过传入数据去迭代和优化函数, 使函数能够完成从输入空间映射到输出空间的功能, 而由于这一个函数过于复杂, 我们只好通过堆砌基本算子 (卷积) 来模拟这个函数, 显然我们还需要让卷积核的数值能够正确地反映出函数的功能, 我们需要根据数据来学习核调整卷积核, 通过从最后一层输出使用链式法则回溯获取每一个算子的梯度, 由于我们使用损失函数来衡量这个函数与真实映射之间的距离, 我们希望这个函数取得更小的值, 而梯度是方向导数在该方向上能取得最大值, 梯度是引导函数更大的方向, 为了让函数更小我们需要让函数参数减去梯度, 这就是深度学习的关键, 梯度下降法。

**梯度计算**, 为了简便计算, 我们省略了 C (channel) 维度, 设输入的特征图和图像为 , 卷积核为 , 输出为 , 损失函数为 , 我们需要知道的是损失函数 对卷积核 的梯度, 根据链式法则有:

(4)

由于损失函数 的公式已知, 所以 已知, 所以只需要计算 就好了, 根据 (1) 提供的卷积前向传播的计算公式, 我们可以得到:

(5)

我们首先考虑 这一组下标对应的卷积核参数:

(6)

对于 这一元素, 对于卷积核参数 的梯度如上, 而对于整个输出 , 对于卷积核参数 应该是一个矩阵, 而后我们再考虑 对 的梯度:

(7)

这样我们就得到了损失函数 对卷积核 的梯度, 而后我们遵循如下公式来更新卷积核 , 即用卷积核减去其梯度来达到使函数往负梯度方向下降的目的:

(8)

## 2.2 池化

### 2.2.1 前向传播

池化操作实际上有两个目的, 其一, 是对数据进行降维, 缩小尺寸, 其二, 是通过缩小尺寸来扩大感受野。所谓感受野就是指中间特征映射到输入空间, 所能够影响的图像区域范围, 感受野越大, 那么中间特征越能够全局综合地感知有效信息, 所以, 大感受野是重要的。对于池化来说, 池化能够将多个特征信息像素压缩成一个特征信息像素, 那么此时单个特征信息像素映射到输入空间影响范围就会更大, 这就是池化能够扩大感受野的原因。

池化主要分为两种, 一种是最大池化 (max pooling), 另一种是平均池化 (average pooling), 首先我们来看池化计算的示意图:

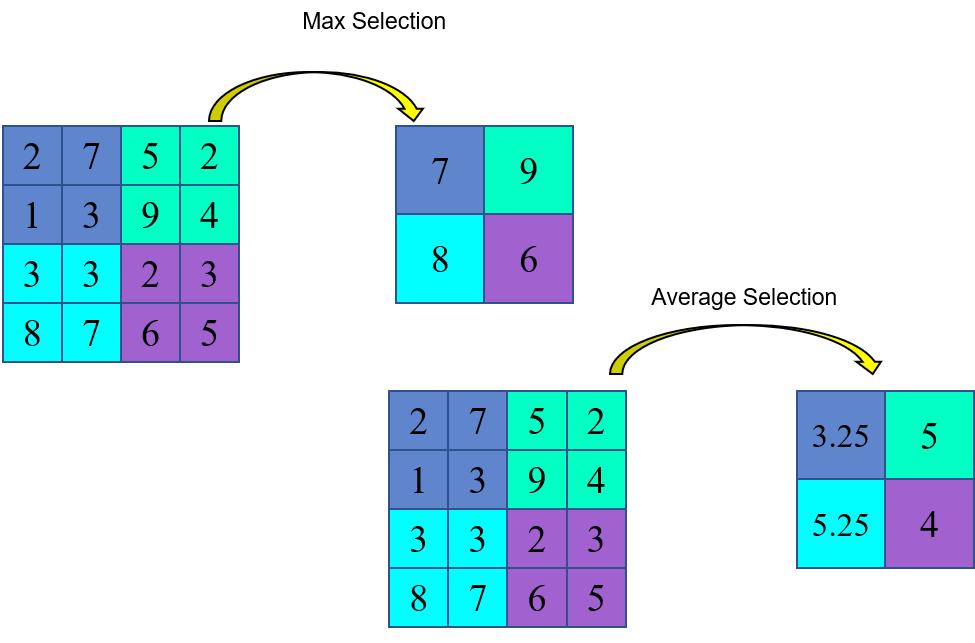


图3. 最大池化和平均池化的前向传播示意图

上图中分别演示了 2x2 池化域的最大池化和平均池化, 从上图中我们可以清晰的发现, 最大池化则是对池化域内的元素取最大值, 而平均池化则是对池化域内的元素取平均值。

在大致介绍了池化的计算细节之后, 我们再介绍一下关于池化的参数, 在此之前我们都默认了我们的池化域为 2x2, 且池化域的滑动步长为 2, 实际上池化并不止这一种固定模式, 池化可以通过超参数设定而变动计算模式, 池化大概有以下参数: 池化域大小 (kernel size), 池化步长 (stride), 填充 (padding), 根据这三个参数我们可以控制池化前后的尺寸变化, 这里给出尺寸变化公式:

(9)

(10)

其中 , 为输出特征图或者图像的尺寸, 而 , 为输入特征图的尺寸, 代表填充 (padding) 大小, 代表池化核大小 (kernel size), 代表池化核滑动步长 (stride), “” 代表向下取整操作。

### 2.2.2 反向传播

池化层作为卷积神经网络中的一环, 在使用梯度下降对卷积神经网络进行优化的时候, 梯度会根据链式法则逐渐传播, 池化也是卷积神经网络的算子之一, 梯度的反向传播会经过池化, 所以我们有必要了解一下池化的梯度计算, 池化的梯度根据池化的计算模式不同而不同, 简单来说就是, 最大池化与平均池化有着不同的梯度计算, 细节如下图:

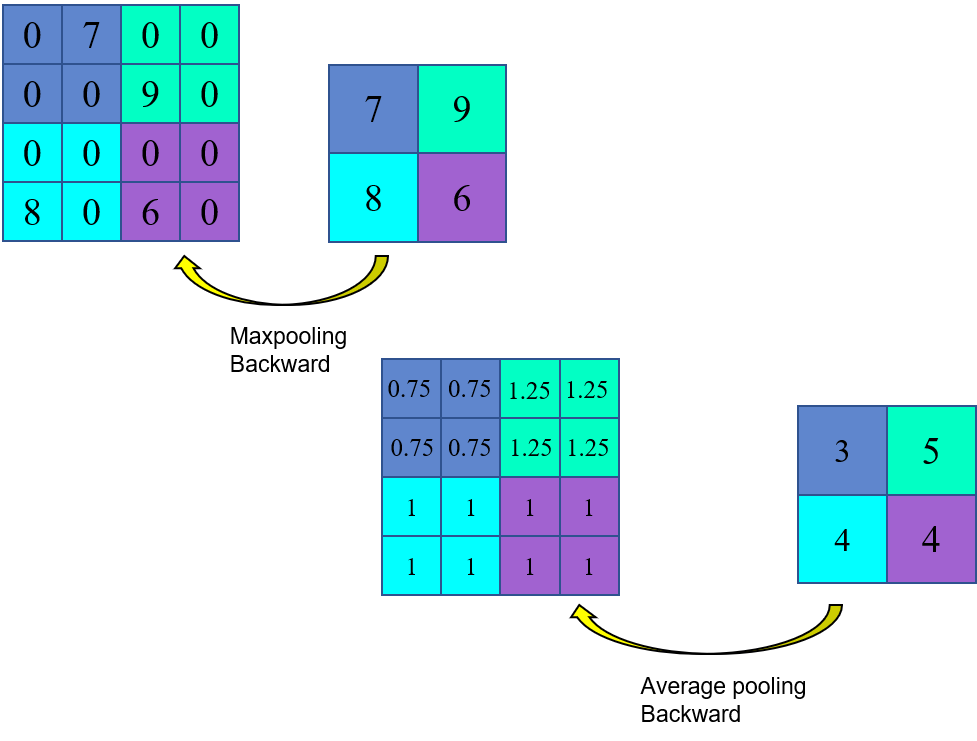


图4. 池化层的反向传播过程

由上图我们可以假设池化层的输入为 , 池化层的输出为 (假设池化层的池化核大小为 2, 池化核滑动步长为 2), 那么最大池化则遵守以下反向传播规则:

(11)

而平均池化则满足以下反向传播规则:

(12)’

## 2.3 激活函数

分析之前的卷积和池化算子, 我们可以发现, 单层的卷积和单层的池化都只能引入线性的映射, 只有再堆叠卷积和池化之后才能够引入非线性, 这也可以部分解释为什么越深的网络效果越好。本质上卷积神经网络通过梯度下降就是为了拟合那个能够将输入空间变换到输出空间的映射, 考虑常规的二维数据分布, 我们可以非常显然地发现, 非线性函数比起线性函数的拟合能力更强:

图表, 散点图

描述已自动生成

图5. y=x 与 y=x^2 对于在第一象限的二维数据分布的拟合情况

为了增强单层卷积或池化的表征能力, 引入激活函数这一模型, 来处理卷积等算子的输出引入非线性, 增强单层的拟合能力, 这就是激活函数的主要目的。

而后我们介绍一下比较经典的激活函数, 这里需要注意的是由于不同的激活函数计算模式不同, 在不断的探索中, 研究者们发现不同的激活函数有着不同的功能, 需要正确使用激活函数, 激活函数才能起到它本身的作用:

SoftMax 函数, SoftMax 能够将数值放缩到 0-1 之间, SoftMax 函数往往作为分类器, 在卷积神经网络的输出阶段将特征向量处理成分类概率, 其计算模式如下:

(13)

Sigmoid 函数, Sigmoid 和 SoftMax 的功能相近, Sigmoid 也能够将数值放缩到 0-1 之间, 所以也能够将特征向量处理成分类概率, 其计算模式如下:

(14)

然而 Sigmoid 函数存在一些问题, Sigmoid 只对在零点附近的数值敏感, 这就导致了当数值偏离零点, 梯度可能会被 Sigmoid 的特性抹除, 从而引发梯度消失的现象。

ReLU 函数, ReLU 函数一般不作为输出的激活函数, 其主要用于卷积神经完了过的中间层, 用来加速网络收敛, 其计算模式如下:

(15)

ReLU 能够抑制负值, 这样的好处就是使得网络只对正值敏感, 而不需要考虑负值, 这缩减了网络的优化空间, 从而使得网络能够更快收敛。

## 2.4 全连接层

首先我们先观察一个卷积神经网络样例:

图示, 工程绘图

描述已自动生成

图6. AlexNet 网络的架构示意图

我们可以发现全连接层主要被放置在最后部分, AlexNet [19] 是一个分类网络, 所以全连接层放置在最后部分的目的在于将之前卷积提取得到的特征转化为分类概率。简单来说, 全连接层在网络当中一般担任分类器的作用。全连接层有其特殊性, 卷积, 池化等算子其主要的目的在于从输入图像中提取特征, 将输入空间变换到特征空间, 而全连接层则将特征空间的数据转化到输出空间, 相当于是解码器的作用, 由于 AlexNet 相对来说是比较老的技术, 其实在较新的卷积神经网络中, 研究者们一般使用 1x1 卷积来替代全连接层, 这是因为全连接层要求将特征图打散为特征向量, 这对于一些需要保持特征图形状的计算机视觉下游任务来说十分不便, 所以通过 1x1 卷积来替代 全连接层。实际上 1x1 卷积就是能够跨通道进行广播计算的全连接层, 通过用卷积来替代全连接层也能够保持整个卷积神经网络的计算一致性。

### 2.4.1 前向传播

全连接层本质上是对特征向量的维度进行放缩, 也是对于信息的一种重分配, 对于全连接层的输出向量, 输出向量的任一维度都是输入向量所有维度的加权和, 输出向量的每一个点都包含了输入向量的所有信息, 这就是全连接层的 “全” 的含义。

我们假设输入特征向量 , 而全连接层的层权重为 , 则输出特征向量为 , 计算公式如下:

(16)

根据上式, 我们可以很轻松的发现, 实际上全连接层就是输入特征向量与层权重的矩阵相乘运算:

(17)

### 2.4.2 反向传播

全连接层也是卷积神经网络中的一部分, 那么全连接层自然也绕不开梯度的计算, 所以我们对全连接层的反向传播做简单的推导, 我们以 (17) 为基准来进行推导, 首先假设 已经得到, 此时我们需要知道 来对全连接层的层参数进行更新, 根据链式法则有:

(18)

对 我们先考虑 , 则有:

(19)

而后我们考虑 , 则有:

(20)

而后变换 , 就可以得到全连接层层参数的梯度 , 而后使用梯度下降来更新层参数:

(21)

## 2.5 批正则化

正则化对于神经网络是一项非常重要的技术, 例如, 输入的时候如果不对图像进行正则化的话, 卷积神经网络将会倾向于去拟合和学习输入图像的像素值与结果之间的关系, 那么让图像乘以某个系数放缩之后, 图像的具体像素值发生了变化, 由于卷积神经网络对像素真值敏感就可能无法得到正确的结果, 如果引入正则化的话, 卷积神经网络将会倾向于去拟合和学习输入图像的像素值分布与结果之间的关系, 这样的学习模式相较于之前更为鲁棒。

研究者们发现了正则化的益处, 于是便思考, 正则化是对于卷积神经网络输入的处理, 那对于卷积神经网络中间特征图传递的时候呢？为了使得在卷积神经网络的中间特征图还能够维持这种正则化的益处, 研究者们提出了批正则化。实际上, 批正则化的引入还有更加深层的原因, 神经网络之所以训练比较困难, 主要是因为神经网络有着相当深的计算层, 对于每一计算层的参数随着梯度下降法更新, 都会导致下一层的输入数据分布发生变化, 通过层层叠加, 深层的输入分布变化会非常剧烈, 从而导致深层需要在向着负梯度方向优化的同时, 还必须兼顾浅层的参数导致的分布变化, 这一现象被称为 Internal Covariate Shift, 批正则化能够使得每一层的输出都被正则化成一个放缩的标准正态分布, 从而使得每一层的输入分布相近, 这就降低了深层的优化压力。

而后我们再来分析一下批正则化的计算细节, 我们假设一个批 (batch) 为 , 包含 个输入图像, , 假设正则化后得到的输出为 , 则有以下操作:

1. 计算这一批数据的均值, 或者当这一批数据并不是数据集的全部时使用滑动平均来计算整个数据集的均值: ;
2. 计算这一批数据的方差, 或者当这一批数据并不是数据集的全部时使用滑动平均来计算整个数据集的方差: ;
3. 使用均值和方差将输入数据的分布处理成标准正态分布: ;
4. 对标准正态分布进行放缩: ;

由于卷积神经网络发展至今, 网络模型往往有着相当的深度, 所以为了避免 Internal Covariate Shift, 或者更直观的梯度爆炸, 收敛太慢等现象, 我们有必要引入 BN 到卷积神经网络当中。

## 2.6 特征提取

对于计算机视觉的各种任务来说, 卷积神经网络最本质的功能是提取特征信息, 而后使用一些解码算子来解码特征信息, 通过解码特征信息, 根据任务的不同定义, 可以得到不同的输出结果。对于计算机视觉三大任务: 分类, 检测, 分割来说, 分类则要求将特征信息解码成分类概率, 而检测则要求将信息解码成目标定位, 分割则要求将其解码成在原始图像上的像素分布。当我们把解码这一过程去掉, 剩下的卷积神经网络, 我们一般将其称为 backbone, 或者 encoder。当卷积神经网络作为 backbone 能够输出更好更高质量的特征, 那么对于下游任务的效果提升也就更有帮助。目前学术界的主要观点是, 卷积神经网络提取三种级别的特征: low-level feature, mid-level feature, high-level feature [20]。其中 low-level feature 指的是卷积神经网络的浅层卷积提取到的特征信息, 一般是比较语义明显的特征, 例如, 边缘, 形态等等; high-level feature 指的是卷积神经网络卷积神经网络的深层卷积提取到的特征信息, 一般是语义比较模糊的特征, 通过观察其实是人不太能够理解的特征信息; mid-level feature 则指的是 high-level feature 和 low-level feature 混合的特征信息, 或者说多尺度的信息, 这一特征信息早期是出现在 FPN [10] 这一目标检测模型和 PSPNet [21] 这一语义分割模型当中。对于卷积神经网络中的特征分布实际上已经有研究者提供了相应的算法, 来分析解构并可视化 [22], 我们首先以一个简单的 8 层卷积神经网络来作为示例:

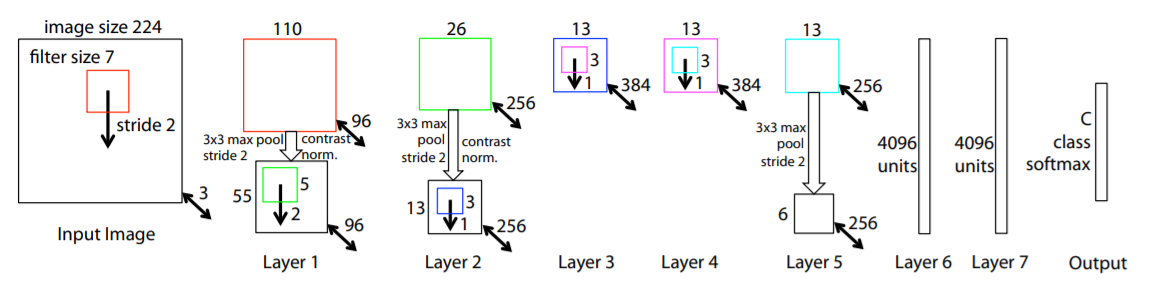


图7. 8 层卷积神经网络样例

而后通过将卷积核的参数转置而后使用反卷积操作将各个卷积层 (layer1, layer2, layer3, layer4, layer5) 的输出特征图映射转化到输入空间, 使得其与原始输入图像尺度保持一致, 将其可视化后如下:

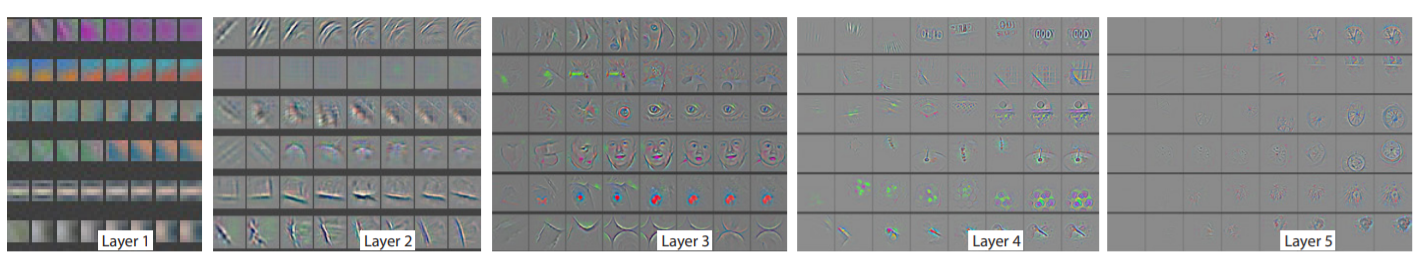


图8. 各层卷积层输出特征图的语义解码图

根据以上的阐述, 我们大致知道了卷积神经网络所能够提取到的特征, 而后我们再分析为什么浅层卷积得到的是 low-level 的特征, 而深层卷积得到的是 high-level 的特征, 由于 mid-level 的特征实际上是 high-level 和 low-level 的叠加, 所以我们在此就不再讨论。回到之前我们有提到的感受野的概念, 感受野指的是特征图上的特征信息点在输入空间上的影响范围, 更加直观的感受野获取方式可见图:

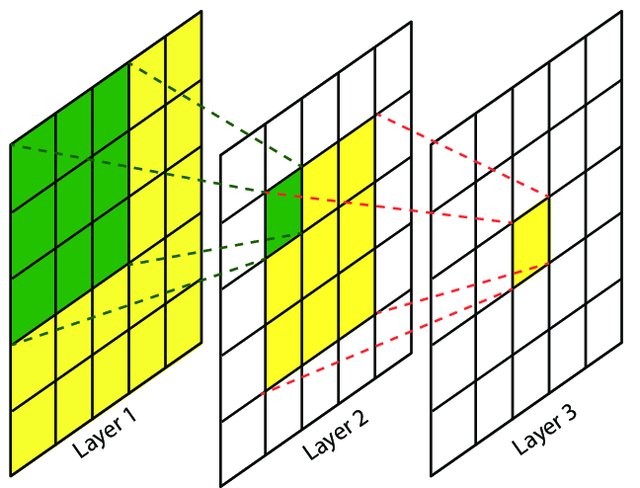


图9. 感受野通过卷积传播的示意图

能够使得像素之间发生交互的算子都能够使得感受野发生扩充, 例如, 卷积能够在卷积核大小内让像素进行交互, 池化能够在池化域大小内让像素进行交互。卷积与池化这两个算子, 都能够通过各自的核来使得感受野扩张, 由于卷积神经网络中的主要算子是卷积, 所以我们主要讨论卷积。卷积神经网络通过堆叠卷积来获得更大的感受野, 感受野的扩大是非常重要的, 这也是为什么卷积神经网络越深往往就能够取得更好的原因之一 [23], 虽然大量堆叠卷积可能会导致网络退化的问题, 不过这不在我们的讨论范畴。在浅层卷积由于感受野较小, 所以其往往只能获取到输入图像的一些局部特征, 这是因为在输入空间浅层卷积输出的特征图所能够影响的范围有限, 所以只能让有限范围的像素发生语义交互, 从而只能得到一些局部信息, 局部信息往往会是一些边缘, 轮廓等语义等级比较低的信息; 深层卷积则能够在映射到输入空间后有较大的对于原始图像的影响范围, 能够发生大范围的像素语义交互, 这使得网络能够获取一些更加全局的信息, 例如, 氛围, 物体的相对关系等等:

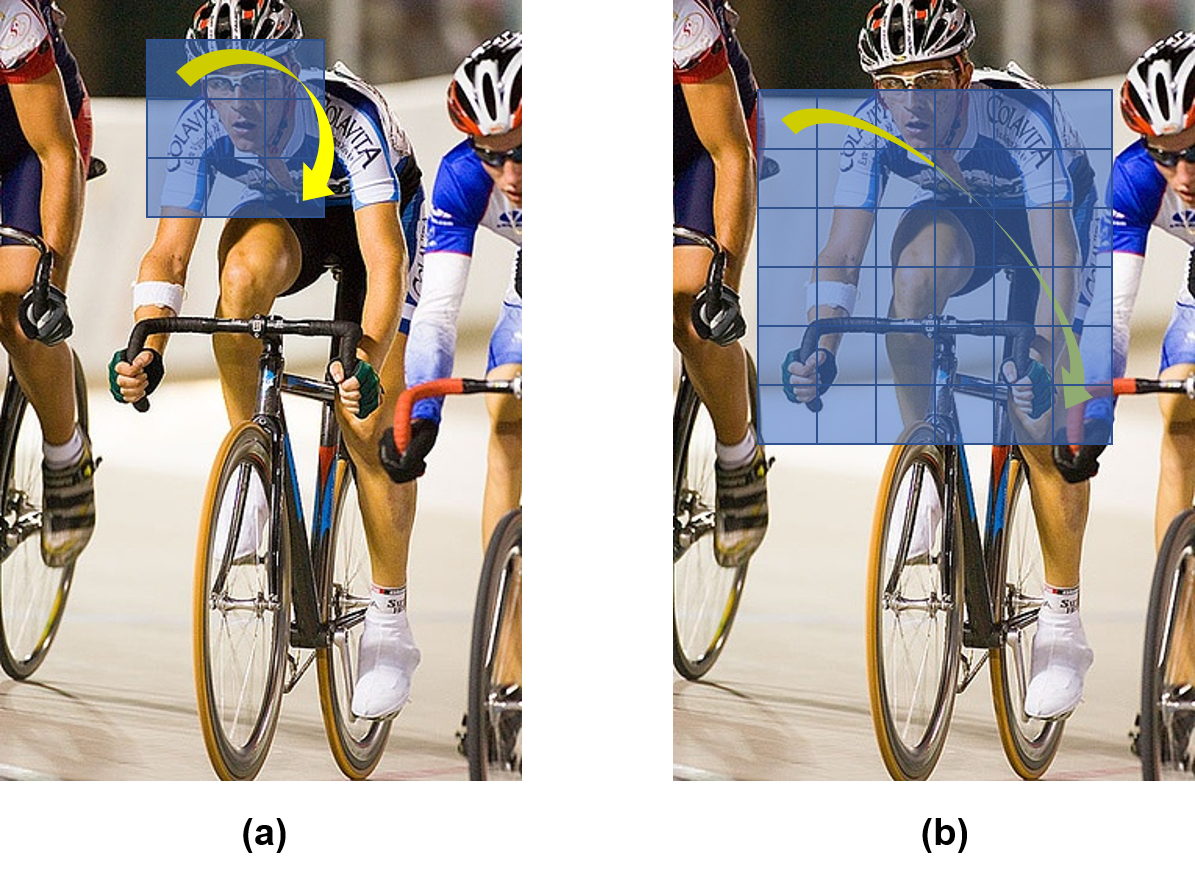


图10. (a) 浅层卷积导致在输入空间的小范围像素语义交互; (b) 深层卷积导致在输入空间的大范围像素语义交互;

在卷积神经网络中正是浅层卷积和深层卷积这两种特性的卷积, 使得高深度的卷积神经网络能够有效地提取对于计算机视觉下游任务有效的特征, 这也是卷积神经网络的意义所在。

# 目标检测

## 任务定义

检测任务是计算机视觉领域非常重要的任务, 其主要目的在于对图像或者视频上的物体框选出来, 并且分析得到物体的类别, 随着互联网, 深度学习, 硬件设备的快速发展, 一方面使得人类社会产生了大量的视频和图像, 另一方面也使得对于视频和图像的处理技术变得越来越先进。前者使得计算机视觉技术在人类社会中变得越来越重要, 后者则使得计算机视觉发展更加成熟, 更能够解决人类社会中的一些实际的视觉问题。在过去, 目标检测相关算法在速度上和精度上都远达不到可以落地投入生产环境的底部, 发展到现在, 目标检测已经与生活中的多个领域相结合, 并且还在持续地扩大其影响力, 例如, 智慧安防, 人脸识别, 人流检测, 遥感图像检测, 医学图像病灶检测等等。

首先我们对于常规的目标检测标注形式进行分析, 首先我们不考虑 anchor-free 的情况。目标检测中用 bounding box 来表征一个物体, bounding box 有两种表示风格, 一种是 voc 风格或 coco 风格, 通过 来表示 bounding box, 其中 , 是 bounding box 左上角角点的坐标, , 是 bounding box 右下角角点的坐标; 另一种则是 yolo 风格, 通过 (, , , ) 来表示 bounding box, 其中 , 代表 bounding box 中心点坐标, 代表 bounding box 的宽, 代表 bounding box 的高。

一群人正在骑自行车

描述已自动生成

图11. VOC 风格的 bounding box 标注 (左); YOLO 风格的 bounding box 标注 (右);

目标检测主要有两个任务, 其一是检出物体, 得到物体的定位, 其二是识别物体, 得到物体的类别, 目标检测模型根据是否整合这两个任务到单一模型, 还是将这两个任务分配到两个模型被划分成 one-stage 检测器和 two-stage 检测器。宏观来看, two-stage 检测器在精度上相较于 one-stage 检测器应该会更好, 因为 two-stage 检测器, 每一个模型需要承担的任务比较简单, 所以训练和优化的压力更小, 但是由于被拆分成了两个阶段串行, 速度一定会大幅度下降; 对于 one-stage 检测器, 其优势在于能够同时对两个任务进行端到端的优化, 而且由于能够对两个任务并行计算, 速度上也会比 two-stage 网络快不少, 不过此时模型的优化压力比较大, 最终精度是不如 two-stage 检测器的。为了去探究 one-stage 检测器和 two-stage 检测器的优劣, 我们将对 one-stage 检测器和 two-stage 检测器做一些细节的阐述。

## 3.2 非极大值抑制

在介绍检测器之前, 我们都需要了解一个操作, 这个操作在目标检测网络中非常重要, 无论是在 two-stage 检测器, 还是在 one-stage 检测器, 在最后网络的输出都会给出大量的 bounding box, 往往 bounding box 的数量会大于图像中物体的数量, 此时就会有一个物体对应多个 bounding box 的情况发生, 为了缓解这样的现象, 我们需要使用非极大值抑制操作来去除多余的框, 此外对于检测器来说非极大值抑制这个操作都是极为耗时的操作, 所以也有研究者致力于研究不需要非极大值抑制地目标检测模型。

蓝色的汽车

描述已自动生成

图12. 非极大值抑制对于重复的 bounding box 的去除效果

在讲解非极大值抑制算法细节之前, 我们需要阐明两个概念: 交并比 (IoU) 和置信度。交并比是一个评价指标, 用于衡量两个 bounding box 之间的交叠程度, 交并比的计算模式如下:

(22)

而后我们介绍置信度这一概念, 我们在通过检测器回归得到 bounding box 的坐标的同时, 我们还会同时回归得到置信度, 置信度对于非极大值抑制算法来说非常重要, 置信度代表的是这个框里包含一个物体的正确程度, 一般来说置信度越高就越说明这个 bounding box 正确地框选了物体。

而后我们对非极大值抑制这一算法进行阐述, 主要分为以下步骤, 假设有 个 bounding box, 包含有这些 bounding box 的集合为 , 其中第 个 bounding box 对应的置信度是 , 非极大值抑制的输出集合为 , 初始状态下为空集:

1. 我们根据置信度 对集合 的 bounding box 进行排序;
2. 我们挑选出置信度最高的个体 , 将其从集合 中去除并放入集合 中;
3. 而后我们让集合 中剩余的 与 计算交并比 (IoU), 当 与 之间的交并比大于某个设定的阈值的时候就将 从集合 去除;
4. 如果集合 已经为空集那么就输出 集合 , 否则跳转到 (1);

通过适当调整交并比的阈值, 非极大值抑制能够很好地去除重复的 bounding box。

## 3.3 two-stage 检测器

之所以要先对 two-stage 检测器进行阐述, 是因为 two-stage 检测器出现得更早, two-stage 检测器的两个 stage 分别是定位和分类, 这在之前已经提到过, 具体来说, 第一个 stage 是得到一系列可能出现物体的候选框 (proposal region), 此时那些不包含物体的图像区域被抑制; 第二个 stage 我们需要遍历所有的候选框 (proposal region), 分析各个候选框是否包含物体, 再分析候选框所包含的物体。

我们以经典的 R-CNN 系列来分析 two-stage 检测器的细节, 首先先看一下最初的 R-CNN 结构:

图示

描述已自动生成

图13. R-CNN 的大致结构: 使用 selective search 算法来提取 region proposals, 将 reigion proposals 放缩到 CNN 的输入尺寸, 而后使用 CNN 来提取特征, 并将其用于 region proposals 的分类。

R-CNN 首先使用 selective search 来得到一系列的候选框 (region proposals), 虽然 selective search 算法致力于提取尽可能包含物体的候选框, 但实际上, 并不能保证所有的候选框都能够包含物体, 而后使用放缩操作将候选框处理成同一尺寸以便输入后续的 CNN 提取特征, 这里针对 CNN 输出的特征, R-CNN 使用 SVM 对齐进行分类, 其实 R-CNN 中还有一个非常重要的操作: bounding box regression, 考虑到 selective search 算法提取得到的候选框可能无法正确反映物体位置, 为了减小定位误差, R-CNN 通过一个回归模型来对 CNN 提取到的特征信息做回归得到一些放缩量, 使用这些放缩量对候选框进行调整, 假设对于某一个真实框 , 候选框的原始值为 , 调整后的候选框为 , 有以下映射关系:

(22)

(23)

(24)

(25)

就是之前提到的放缩量, CNN 提取得到的特征向量通过全连接层回归得到四个参数, 即这四个放缩量。

对于 R-CNN 来说, 其完整结构可以被整理如下:

图示

描述已自动生成

图14. R-CNN 的完整结构

而后我们再分析 Fast R-CNN, 我们观察 Fast R-CNN 的大致结构:

图示

描述已自动生成

图15. Fast R-CNN 的大致结构

首先 Fast R-CNN 将 SPPNet 的 ROI Pooling 模块引入了 R-CNN 中, 这就使得候选框不再需要缩放到同一大小而输入 CNN 当中, 通过 ROI Pooling, 任意大小的候选框都能够处理成同一大小的特征图而输入 CNN 当中, 此外, Fast R-CNN 虽然依旧使用 selective search 操作来得到候选框, 但是获得候选框的定位后, Fast R-CNN 不再对每一个候选框都取出对应的图像而后放入 CNN 中提取特征信息, 而是首先直接使用 CNN 来提取整张图像的特征信息得到特征图, 而后再使用候选框去除对应位置的特征, 这一操作大大减少了对原始图像提取特征信息的重复计算, 而后由于大小不同的特征图被提取出来, 使用 ROI Pooling 将其处理成同样尺寸的特征向量, 而后特征向量被分别送到分类和定位的两个计算器当中, 得到候选框的定位放缩值用于 bounding box 的微调, 分类概率用于分类。

这里我们对 ROI Pooling 为什么能够使得不同大小的特征图输出同样长度的特征向量做一个分析:

日程表

中度可信度描述已自动生成

图16. SPPNet 的 ROI Pooling 操作

ROI Pooling 使用了三个尺度, 分别将输入图像或者特征图划分成 4x4, 2x2, 1x1 的网格, 而后将网格中的每一个格子的内容压缩成一个 256 维度的特征向量, 这样无论输入图像或特征图以什么样的尺寸输入都会获得 4x4, 2x2, 1x1 的网格, 压缩后则是 4x4x256 + 2x2x256 + 1x1x256 维度的特征向量。

最后我们再来分析 Faster R-CNN:

图形用户界面

描述已自动生成

图17. Faster R-CNN 大致结构

发展到 Faster R-CNN 这一阶段, 二阶段检测器就已经变得比较成熟, Faster R-CNN 当中最为重要的就是 RPN 结构, RPN 结构替换了 Selective Search 操作, 来获取候选框, 这使得网络能够直接端到端进行优化, 大大提升了网络的训练效率, 同时也提升了整个网络的推理速度。RPN 的结构如下图:

图示

描述已自动生成

图18. RPN 结构的计算细节图

RPN 网络有两条分支, 其中一条分支用于计算候选框的 position 和 negative 分类, 这主要用于判断候选框是否有正确框选物体, 还有一条分支用于计算候选框的坐标, 这里需要注意的是, Faster R-CNN 中引入了 anchor 的概念, 之所以第一条分支的 1x1 卷积输出的通道数为 18, 是因为对于单个特征点, RPN 将根据 anchor 的数量得到9 个 bounding box (anchor 数为 9), 而每个 bounding box 需要两个通道来代表 positive 和 negative 分类, 所以需要 2x9 个通道, 之所以第二条分支的 1x1 卷积输出通道数为 36, 是因为对于这一条分支将得到 bounding box 的坐标, 单个 bounding box 需要 4 个通道来表达, 而因为 anchor 总共有 9 个, 所以单个特征点也会生成 9 个 bounding box, 这种基于 anchor 和特征图上的特征点来生成 bounding box 的形式大大提升了 two-stage 检测器的精度和速度。

## 3.4 one-stage 检测器

对于 one-stage 检测器, 我们可以把这个单一 stage 的模型整理成如下构造:

图示

描述已自动生成

图19. 一阶段检测器的结构

在 one-stage 检测器中, 模型的不同部分的角色不同, backbone 负责将数据从输入空间转换到特征空间, 从输入图像中提取丰富的特征信息, head 则负责解码这些丰富的特征信息, 将其处理成物体的定位和分类, neck 其实并不是必须的, 当 backbone 的输出与 head 的输入无法直接串联的时候, 我们需要 neck 结构作为中转将 backbone 的输出特征图处理成 head 可以接受的输入形式, 这就是 neck 的用处。

对于 backbone 这一结构, 是所有计算机视觉下游任务都必须要包含的结构, 一般目标检测任务中的 backbone 都会直接继承于分类任务的相关网络, 分类任务可以说是最为简单的计算机视觉任务, 因为网络的输出可以不需要经过解码就能够直接作为结果, 所以我们一般认为, 当网络在分类任务上效果越好时, 其对于图像的表征能力, 其对于输入空间和特征空间的转换能力, 其对于特征的提取能力就越好, 得到的特征信息质量就会越高, 从而越有利于计算机视觉下游任务。所以计算机视觉下游任务一般都会选择强力的分类网络作为 backbone, 目标检测自然不例外。

我们以 YOLOv3 为例来分析 one-stage 检测器技术, YOLO 的结构如下图, 由于 backbone 选择的多样性, 我们将 backbone 的具体结构省略:

图片包含 应用程序

描述已自动生成

图20. Yolov3 的网络的结构细节 (Bakcbone 省略)

Yolov3 的结构可以归纳为 Backbone-Head 结构, 其中 backbone 的选择, Yolov3 提供了 darknet-53, 这是作者专门为 Yolov3 构建的分类网络, 作者使用 darknet-53 在 ImageNet 数据集上进行了预训练, 而后将其作为 Yolov3 的 backbone 用于从图像当中提取特征, backbone 的设计不是重点, Yolov3 的关键在于其 head 的设计, Yolov3 对于 head 的设计也可以说是 one-stage 检测器的规范模板, 在 Yolo 之后的 one-stage 检测器对于 head 的设计都或多或少受到了 yolo 的影响。

Yolov3 有三个 head, 之所以要这样设计是因为为了适应检测目标的不同大小, Yolov3 将图像划分成 grid, 对于每一个 head, grid 的个数不同, grid 的个数取决于输出的特征图分辨率, 例如, tiny head 最后输出的特征图为 7x7 的尺寸, 那么就会把图像划分成 7x7 个 grid。输出特征图分辨率越小, 单个 grid 的尺度越大, 就越能够捕捉到大目标, 单个 grid 的尺度越小, 能越能够捕捉到小目标。那么各个 head 的特征图如何解码成 bounding box 的坐标呢？这里, Yolov3 借鉴了 R-CNN 系列的 bounding box regression 和 anchors 的思想:

图示

描述已自动生成

图21. Yolov3 中的 bounding box regression

首先, 这里对 255 这一输出通道数做一个解释, 每一个 head 输出的特征图, 上的每一个特征点对应着三个 bounding box, 所以 Yolov3 的输出构成是这样的: 3\*(bounding box 坐标, 置信度, 分类概率), 由于原始的 Yolov3 主要是在 coco 数据集上训练测试, 所以分类概率所占的通道数是 80, 而置信度所占的通道数为 1, bounding box 坐标所占的通道数为 4, 最终为 3\*(80+1+4), 即 255。置信度这个属性在非极大值抑制章节中已经做过解释, 分类概率这个属性 Yolov3 没有做一些处理, 就只是单纯的分类概率, 而对于 bounding box 坐标和长宽, Yolov3 根据 R-CNN 中 bounding box regression 和 anchor 的思路, head 输出的实际上是 bounding box 的参数化坐标, 或者说是 bounding box 的放缩量, 要得到 bounding box 在图像坐标系下的坐标, 需要一系列的运算公式, 为了表达运算公式, 我们设 Yolov3 的 head 关于 bounding box 的输出为 , 而 anchor 的长宽为 , bounding box 在图像坐标系下的坐标和长宽为 , , 为 bounding box 所在 grid 的左上角的坐标, 则有以下关系 (图21):

(26)

(27)

(28)

(29)

其中 代表 sigmoid 函数, 每一个 head 的每一个特征点要输出三个 bounding box, 三个 bounding box 由三个 anchors 放缩而来, 考虑到有三组 head, 所以总共有 9 个 anchor, 就以上述的输出设定来看, tiny head, middle head, large head 分别能够输出 7\*7\*3, 14\*14\*3, 28\*28\*3 个 bounding box, 从总和来看, 最多能输出 3087 个 bounding box。

## 3.4 anchor-free 检测器

对于 anchor free 的检测器, 我们也简单地分析介绍, 因为 anchor-free 检测器是目标检测中一种非常重要的技术, 也是今后目标检测发展的趋势。

我们以 FCOS 为例, 来阐述 anchor-free 检测器, FCOS 主要有三大特点: 逐像素的回归预测, 多尺度特征, center-ness 策略抑制低质量 bounding box, FCOS 在后阶段, 只使用了非极大值抑制, 而舍弃了 anchor 等操作, 在保持和 anchor-based one-stage 检测器精度对齐的基础上, 速度会比 anchor-based one-stage 检测器更快, FCOS 网络架构示意图如下:

图示, 形状, 多边形

描述已自动生成

图22. FCOS 的网络架构

FCOS 的 backbone 以及 Feature Pyramid neck 与 anchor-based 检测网络没有什么不同, 而对于 head 的设计则是 FCOS 这一 anchor-free 检测器与 anchor-based 检测器区分开来的关键。Head 将会输出三条分支, 一条分支是分类分支, 将会输出结果特征图上各个特征点所对应的 bounding box 的分类概率, 一条分支是回归分支, 将会输出结果特征图上各个特征点对应的 bounding box 的相关表征, 还有一条分支则是 center-ness 分支, center-ness 将会提供一个额外的约束来优化检测器, 使得整个模型能够获得更加精准的 bounding box。

回归分支, 回归分支就是最能够体现 FCOS anchor-free 分支的要点所在, FCOS 在回归分支输出的特征图上的每一个特征点都代表了一个 bounding box, 而后FCOS 使用了一种新的表征 bounding box 的方式:

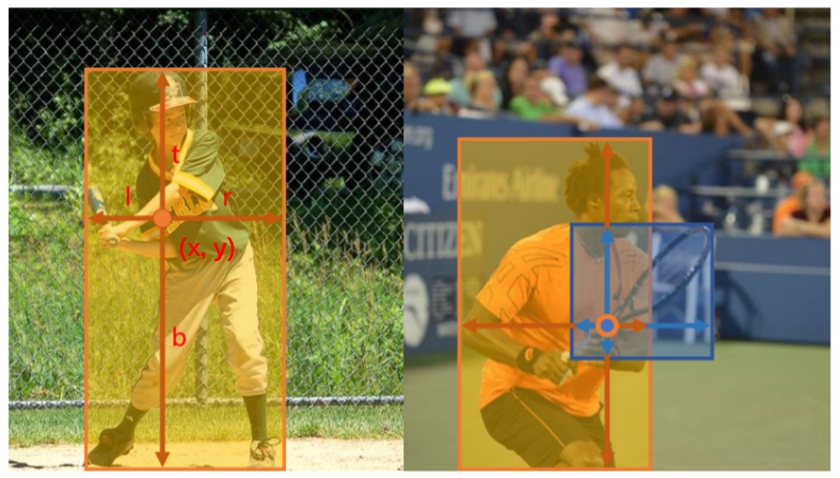


图23. FCOS 表征 bounding box 的方式

我们可以看到, FCOS 使用了一个中心点 , 和中心点到 bounding box 四条边的距离: , 我们使用数学公式对其进行表达, 首先我们先解释一下 FCOS 如何定义 bounding box 的中心点, 中心点的定义并不取决于网络的输出, 而取决于中心点对应的特征点在特征图上的坐标, 设回归输出特征图为 , 对于输出特征图上的某个特征点则为 , 其在特征图上的坐标为 , 输入图像为 , 其中尺度变换比例 , 那么该特征点对应的 bounding box 在输入图像上的坐标 为 (“” 代表向下取整操作):

(30)

(31)

而后对于回归分支的输出 , 假设 为 bounding box 左上角在图像中的坐标, 为 bounding box 右下角在图像中的坐标, 我们可以得到以下关系:

(32)

(33)

(34)

(35)

以上就是关于 FCOS 如何表征 bounding box 的细节 -- 参数中心点 , 偏移量 的如何获取。遍历 FCOS 的实现过程, 我们可以发现 FCOS 对于 bounding box 的表征方式, 避开了对于 anchor 的依赖, 而这就是 anchor free 的关键所在。

对于 center-ness 分支, 这个分支主要作用在于为网络增加一个额外的任务, 增加一个额外的损失函数, 提供一个额外的约束来优化整个网络, 使得网络能够预测得到更加精准的 bounding box, 我们分析一下其主要的计算细节, 设此时某个特征点的回归分支的输出为 , 对应特征点的 center-ness 分支输出为 :

(36)

(37)

center-ness 分支与分类分支共用一组特征向量, 该分支将特征向量回归得到一个 0-1 的值, 网络通过二分类交叉熵 (BCE) 希望这个值能够与 center-ness 有着同样的分布。

# 手写数字及写字人识别实验过程及其结果

（各章标题：黑体，小二号，居中，单倍行距，段前、段后各0.5行；章节序号与标题之间空一字符）

## 手写数字识别实验

（各节一级标题：黑体，小三号，居左，单倍行距，段前、段后各0.5行）

### 样本简介

（各节二级标题：黑体，四号，居左，单倍行距，段前、段后各0.5行）

（正文：1.5倍行距；中文：宋体，小四号，每段首行空2个汉字；字母和阿拉伯数字：Times New Roman字体，小四号）

本论文的手写数字识别实验当中所用的样本分为两类，一类是训练样本集，另一类是测试样本集。

实验当中的训练样本集采用的是手写数字MNIST数据库。这个数据库当中包含训练集样本60000个样例和测试集样本10000个样例。MNIST数据库当中的数字样本已经全部大小归一化灰度化并且集中到同一个固定大小的图像当中。该数据库包括MST的SD-1和SD-3数据库，当中包含一系列的二级制的手写数字图像。其中SD-1的收集者来源是某高中的在校学生，而SD-3是由人口调查局员工收集的。则我们的训练样本集也就是MNIST当中的训练样本集有30000个样本来自SD-3，而另外30000个样本来自SD-1。这60000个训练样本分别来自约250个采集者。

### Writer Depend类数字识别实验

#### ABCvsA数字识别实验

（各节三级标题：黑体，小四号，居左，单倍行距，段前、段后各0.5行）

实验内容：以A写字人、B写字人和C写字人，合计3000个数字0到9的数字图像数据为训练样本集。A写字人的1000个数字0到9的数字图像数据为测试样本集。学习率为1，单次训练样本数为10个，共训练40次。若识别所得数字与给定的标签匹配，则视为正确；不匹配则视为错误。

表4-1 ABCvsA数字识别实验结果

（表的标题：位于表的上方，一般居中，宋体，五号；表的序号：按章编排，如此表为第四章第一个表，则序号为“表4-1”，序号与文字描述之间空一格）

（表格不加左、右列线；表内数字空缺的格内加“—”字线）

（表中文字：宋体，五号）

|  |  |  |  |
| --- | --- | --- | --- |
| 训练样本 | ABC | 样本个数 | 3000 |
| 测试样本 | A | 样本个数 | 1000 |
| 训练次数 | — | 单次训练样本数 | 10 |
| 学习率 | 1 | 正确率 | 99.50% |

#### ABCvsABC数字识别实验

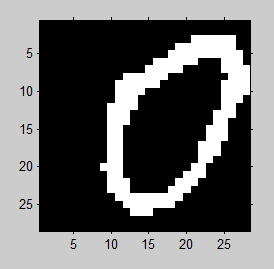
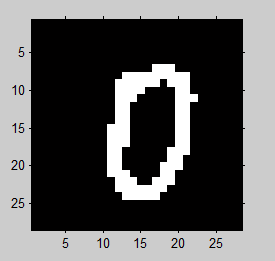
实验内容：以A写字人、B写字人和C写字人，合计3000个数字0到9的数字图像数据为总样本集。在总样本集当中随机抽取2400个为训练样本集，余下的600个为测试样本集。学习率为1，单次训练样本数为10个，共训练40次。若识别所得数字与给定的标签匹配，则视为正确；不匹配则视为错误。

表4-2 ABCvsABC数字识别实验结果

|  |  |  |  |
| --- | --- | --- | --- |
| 训练样本 | ABC | 样本个数 | 2400 |
| 测试样本 | ABC | 样本个数 | 600 |
| 训练次数 | 40 | 单次训练样本数 | 10 |
| 学习率 | 1 | 正确率 | 92.00% |

### Writer Depend类数字识别实验结果分析

下面我们选取Writer Depend类数字识别实验当中的两个典型的例子ABCvsA数字识别实验以及MNIST&ABCvsA数字识别实验的结果做详细分析。我们从ABCvsA数字识别实验中的训练样本集和测试样本集的手写数字图像样本集当中分别随机抽取一幅图像如图4-1所示。

1. 实验训练集 b)实验测试集

图4-1 ABCvsA数字识别实验集

（图的标题：位于图的下方，一般居中，宋体，五号；图的序号：按章编排，如此表为第四章第一个图，则序号为“图4-1”，序号与文字描述之间空一格）

（图中若有分图时，分图号用a)、b)等置于分图之下）

下面我们对上述的训练集和测试集进行40次学习率为2，单次训练样本为10的迭代，得到错误率为0.50%，而其中每次训练时的误差值组成的历史误差值画图分析如下：

……

### Writer Independ类数字识别实验

实验内容：以MNIST数据库为训练样本集，共计60000个训练样本。以A写字人合计1000个数字0到9的数字图像数据为测试样本集写字人识别实验

……

### 样本简介

……

### 两位写字人识别实验

#### 单个数字的写字人识别实验

实验内容：以A写字人，合计800个数字5的数字图像数据加上B写字人，合计800个数字5的数字图像数据，共计1600个样本为总样本集。随机选取其中的1200个样本为训练样本集，其余的400个样本为测试样本集。学习率为2，单次训练样本数为10个，共训练30次。若识别所得写字人与给定的标签匹配，则视为正确；不匹配则视为错误。

表4-3 单个数字写字人识别实验结果

|  |  |  |  |
| --- | --- | --- | --- |
| 训练样本 | A5&B5 | 样本个数 | 1200 |
| 测试样本 | A5&B5 | 样本个数 | 400 |
| 训练次数 | 30 | 单次训练样本数 | 10 |
| 学习率 | 2 | 正确率 | 99.75% |

#### 单个数字的写字人识别实验结果分析

……

## 本章小结

……。

# 结论

（总结标题：黑体，小二号，居中，单倍行距，段前、段后各0.5行）

## 论文工作总结

（各节一级标题：黑体，小三号，居左，单倍行距，段前、段后各0.5行）

……

## 工作展望

……

# 参考文献

1. Redmon J, Farhadi A. Yolov3: An incremental improvement[J]. arXiv preprint arXiv:1804.02767, 2018.
2. Deng J, Guo J, Zhou Y, et al. Retinaface: Single-stage dense face localisation in the wild[J]. arXiv preprint arXiv:1905.00641, 2019.
3. Narasimhaswamy S, Wei Z, Wang Y, et al. Contextual attention for hand detection in the wild[C]//Proceedings of the IEEE/CVF International Conference on Computer Vision. 2019: 9567-9576.
4. Girshick R, Donahue J, Darrell T, et al. Rich feature hierarchies for accurate object detection and semantic segmentation[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2014: 580-587.
5. Girshick R. Fast r-cnn[C]//Proceedings of the IEEE international conference on computer vision. 2015: 1440-1448.
6. Ren S, He K, Girshick R, et al. Faster r-cnn: Towards real-time object detection with region proposal networks[J]. arXiv preprint arXiv:1506.01497, 2015.
7. He K, Gkioxari G, Dollár P, et al. Mask r-cnn[C]//Proceedings of the IEEE international conference on computer vision. 2017: 2961-2969.
8. Cai Z, Vasconcelos N. Cascade r-cnn: Delving into high quality object detection[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2018: 6154-6162.
9. He K, Zhang X, Ren S, et al. Spatial pyramid pooling in deep convolutional networks for visual recognition[J]. IEEE transactions on pattern analysis and machine intelligence, 2015, 37(9): 1904-1916.
10. Lin T Y, Dollár P, Girshick R, et al. Feature pyramid networks for object detection[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2017: 2117-2125.
11. Redmon J, Divvala S, Girshick R, et al. You only look once: Unified, real-time object detection[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 779-788.
12. Redmon J, Farhadi A. YOLO9000: better, faster, stronger[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2017: 7263-7271.
13. Liu W, Anguelov D, Erhan D, et al. Ssd: Single shot multibox detector[C]//European conference on computer vision. Springer, Cham, 2016: 21-37.
14. Lin T Y, Goyal P, Girshick R, et al. Focal loss for dense object detection[C]//Proceedings of the IEEE international conference on computer vision. 2017: 2980-2988.
15. Tian Z, Shen C, Chen H, et al. Fcos: Fully convolutional one-stage object detection[C]//Proceedings of the IEEE/CVF International Conference on Computer Vision. 2019: 9627-9636.
16. Dosovitskiy A, Beyer L, Kolesnikov A, et al. An image is worth 16x16 words: Transformers for image recognition at scale[J]. arXiv preprint arXiv:2010.11929, 2020.
17. Zeiler M D, Fergus R. Visualizing and understanding convolutional networks[C]//European conference on computer vision. Springer, Cham, 2014: 818-833.
18. Ding X, Guo Y, Ding G, et al. Acnet: Strengthening the kernel skeletons for powerful cnn via asymmetric convolution blocks[C]//Proceedings of the IEEE/CVF International Conference on Computer Vision. 2019: 1911-1920.
19. Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks[J]. Advances in neural information processing systems, 2012, 25: 1097-1105.
20. He K, Zhang X, Ren S, et al. Deep residual learning for image recognition[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 770-778.
21. Zhao H, Shi J, Qi X, et al. Pyramid scene parsing network[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2017: 2881-2890.
22. Zeiler M D, Fergus R. Visualizing and understanding convolutional networks[C]//European conference on computer vision. Springer, Cham, 2014: 818-833.
23. Szegedy C, Liu W, Jia Y, et al. Going deeper with convolutions[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2015: 1-9.
24. LeCun Y, Bottou L, Bengio Y, et al. Gradient-based learning applied to document recognition[J]. Proc. IEEE, 1998, 86(11): 2278-2324.

期刊文献

［序号］作者．文献题名[J]．刊名,出版年份,卷号(期号)：起-止页码.

1. 刘国钧，陈绍业，王凤翥.图书馆目录[M].北京:高等教育出版社，1957.15-18.

学术著作

［序号］作者．书名[M].出版地:出版社, 出版年: 起-止页码 .

1. Ngiam J, Chen Z, Chia D, et al. Tiled convolutional neural networks[C], Advances in Neural Information Processing Systems. 2010: 1279-1287.

有ISBN号的论文集

［序号］作者．题名[A].主编．论文集名[C]．出版地：出版社,出版年：起-止页码.

1. 田露. 基于多特征数据融合的离线中文笔迹鉴别研究[D]. 河南大学, 2011.
2. 张慧档. 笔迹鉴别方法研究[D]. 郑州大学, 2002.
3. 梁亮. 图像处理技术在笔迹鉴定系统开发过程中的应用与研究[D]. 沈阳工业大学, 2007.
4. 陈先昌. 基于卷积神经网络的深度学习算法与应用研究[D]. 浙江工商大学, 2014.
5. 王强. 基于CNN的字符识别方法研究[D]. 天津师范大学, 2014.

学位论文

［序号］作者．题名[D]．授予单位地：授予单位,年份.

1. 姜锡洲.一种温热外敷药制备方案[P].中国专利:881056073，1989-07-26.

专利文献

［序号］专利所有者．专利题名[P]．专利国别：专利号,发布日期.

1. GB/T 16159-1996，汉语拼音正词法基本规则[S].

技术标准

[序号] 标准代号,标准名称[S].出版地：出版者,出版年

1. 谢希德.创造学习的新思路[N].人民日报，1998-12-25(10).

报纸文章

[序号］作者．题名[N]．报纸名,出版日期(版次)

1. 冯西桥.核反应堆压力管道和压力容器的LBB分析[R].北京:清华大学核能技术设计研究院，1997.

报告

［序号］作者．文献题名[R]．报告地：报告会主办单位,年份

1. 王明亮.关于中国学术期刊标准化数据库系统工程的进展[EB/OL].http://www.cajcd.edu.cn/pub/wml.txt/980810-2.html'1998-08-16/1998-10-04.

电子文献

［序号］作者．电子文献题名[文献类型/载体类型]．文献网址或出处,发表或更新日期/引用日期(任选)

1. Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks[C], Advances in neural information processing systems. 2012: 1097-1105.
2. Zeiler M D, Fergus R. Visualizing and understanding convolutional networks[M], Computer Vision–ECCV 2014. Springer International Publishing, 2014: 818-833.
3. Zeiler M D, Krishnan D, Taylor G W, et al. Deconvolutional networks[C],Proc. CVPR, 2010: 2528-2535.

# 致谢

……