南京農業大學
NANJING AGRICULTURAL UNIVERSITY

数字人文下的汉字处理

目录

CONTENT

- 汉字基本知识
- 汉字编码
- 汉字处理程序
- 篆体字识别

南京農業大學
NANJING AGRICULTURAL UNIVERSITY

汉字基本知识

1、字汇

◆指用汉字编码字符集或者类别指定的汉字集合。字汇本身不涉及编码的概念，仅仅是字符的集合。繁体字、简化字、传承字、异体字等涵盖了汉字字汇的整体内容。

- ●简化字：也称简体字，是中华人民共和国的标准字汇。
- ●繁体字：与简化字对应，可以认为是被简化字所代替的汉字。
- ●传承字：未被简化的汉字，目前使用的规范汉字主要包括简化字和传承字。
- ●异体字：异体字与正体字相对应，指音义和使用功能相同而字形不同于正体字的汉字。

2、字形

◆汉字的外形呈现就是字形，包含了笔画数目、笔画形状、结构方式和笔顺。GB/T 16964.1-1997也将其定义为“一个可以辨认的抽象的图形符号，它不依赖于任何特定的设计”。

◆现行的《通用规范汉字表》由教育部、国家语言文字工作委员会组织制定，是基于《第一批异体字整理表》、《简化字总表》、《现代汉语常用字表》和《现代汉语通用字表》等字表制定的。该表收录三级共8105个汉字，字形依据《印刷通用汉字字形表》确定，即宋体标准字形。

3、字型和字体

◆字型的概念源于现代铅字印刷术，是一整套具有同样样式、尺寸的字形，也就是一整套可用于印刷的铅字，有具体的大小、粗细等。

◆字型按造型方法可分为三类：点阵字型、矢量字型、轮廓字型。

◆字体为具有同一风格的字形，常见的有宋体、仿宋、楷体、黑体、隶书、行书等。矢量和轮廓字型的出现，使得字型和字体的概念边界变得模糊，信息技术中两者逐渐混用。

南京農業大學
NANJING AGRICULTURAL UNIVERSITY

汉字编码

ASCII

在计算机系统中，由美国的标准信息交换码体系所规定的西文字符编码，被称为American Standard Code for Information Interchange（ASCII码）。在ASCII码的所有7位版中共包含了128个字符（27=128），具体由52个大小写英文字母、10个阿拉伯数字、32个运算符和标点符号与34个控制码。ASCII码也有8位的扩展版本，收集了西方文字中的一些特殊字母和符号。

GB2312-80

早期使用的汉字编码字符集。每个字符由两个字节表示，两个字节的码位都是161～254，编码空间为94×94＝8836。GB2312-80的第一个汉字为“啊”，其编码为：176，161。GB2312-80共包含6763个通用汉字，加其他字符共7445个。GB2312-80按字形编码，多音字一码，同音字多码。GB2312-80兼用ASCII码，因此存在跟ASCII字符共同处理的问题：如果传输过程中丢失GB字符的某个字节，就会发生错码。

GB2312-80

通过控制首字节，GB2312-80的对编码区间进行了如下划分：

第1区（首字节161）：标点和符号；第2区（首字节162）：特殊数字；第3区（首字节163）：阿拉伯数字和拉丁字母；第4～5区（首字节164、165）：平假名和片假名；第6区（首字节166）：希腊字母；第7区（首字节167）：斯拉夫字母。第8区（首字节168）：带调元音和注音字母；第9区（首字节169）：制表符；每区最多94个字符，汉字在第10～87区。

Big5

Big5通常也叫做“繁体中文”编码集，是中国台湾地区和港澳地区常用编码集。分为常用字和次常用字，分别按照笔画数和部首来排序。编码空间为：第一字节161～254，第二字节64～126，161～254，共有14758个码位。同样是是94个区，但每个区有94+63位。

GBK

GBK（GB13000）是目前最常见的汉字编码字符集。其编码空间为：第一字节为129～254，第二字节为64～254（缺127），共有23940个码位，其中汉字20907个。GBK兼容GB2312-80的所有汉字（6763个汉字的代码有简单的对应关系），而且在字汇一级支持CJK，涵盖Big5（但代码不一致）。

Unicode

国际标准化组织制定的编码字符集，基本可以容纳所有文字符号。具体实现的编码方案有：UTF（Unicode Transformation Format）8、16、32。

UTF8最常用，是变长字符编码，由1-4字节表示。UTF8的首字节兼容ASCII码，汉字一般由3-4个字节编码。UTF16次常用，是等长字符编码，由2或4字节表示。UTF32直接由4个字节编码，目前还没有真正普及。

南京農業大學
NANJING AGRICULTURAL UNIVERSITY

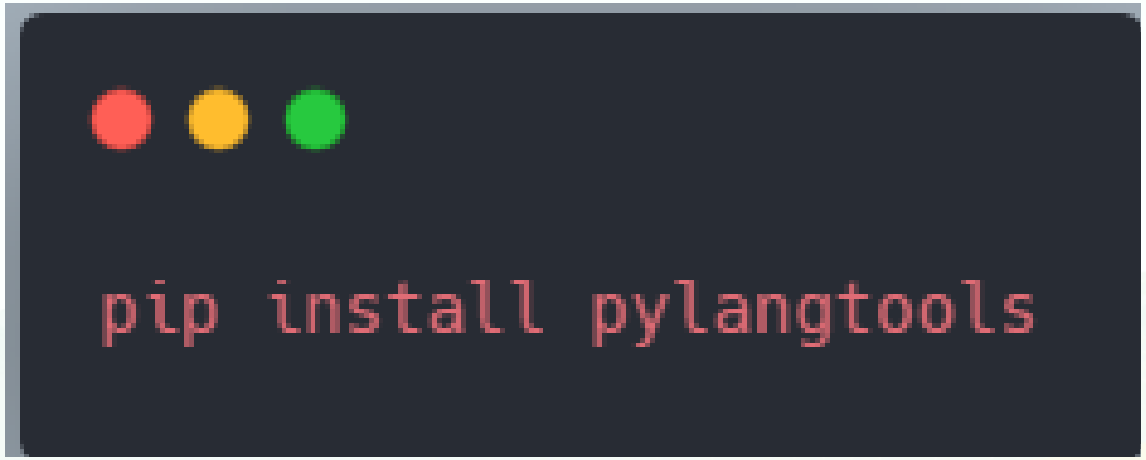
汉字处理程序

繁体简体相互转换

◆方法一：

请确保电脑安装了Python并正确设置了环境变量

打开windows命令提示符，安装所需Python库：



通过Python程序进行繁体简体相互转换

◆繁体转简体

新建Python脚本文件，输入代码：

```
from pylangtools.langconv import Converter
text_t = '欽定四庫全書'
text_s = Converter('zh-hans').convert(text_t)
print('转换后的简体字为：{}'.format(text_s))
```

其中，text_t为待转换繁体文本。

通过Python程序进行繁体简体相互转换

◆简体转繁体

新建Python脚本文件，输入代码：

```
from pylangtools.langconv import Converter
text_s = '钦定四库全书'
text_t = Converter('zh-hant').convert(text_s)
print('转换后的繁体字为：{}'.format(text_t))
```

其中，text_s为待转换繁体文本。

通过Python程序进行繁体简体相互转换

◆方法二：

打开windows命令提示符，安装所需Python库：

通过Python程序进行繁体简体相互转换

◆繁体转简体

新建Python脚本文件，输入代码：

```
import zhconv
text_t = '欽定四庫全書'
text_s = zhconv.convert（text_t, 'zh-hans'）
print（'转换后的简体字为：{}'.format（text_s））
```

其中，text_t为待转换繁体文本。

通过Python程序进行繁体简体相互转换

◆简体转繁体

新建Python脚本文件，输入代码：

```
import zhconv
text_s = '钦定四库全书'
text_t = zhconv.convert(text_s, 'zh-hant')
print('转换后的繁体字为：{}'.format(text_t))
```

其中，text_s为待转换繁体文本。

文本仅保留中英文、数字和符号

新建Python脚本文件，输入以下代码，可使用该函数进行初步的文本数据清洗。

```
import re
def clear_character(sentence):
    #只保留中英文、数字和符号
    pattern = re.compile("[^\u4e00-\u9fa5^,^.^!^a-z^A-Z^0-9]")
    #若只保留中英文和数字，则替换为[^\u4e00-\u9fa5^a-z^A-Z^0-9]
    line=re.sub(pattern,'',sentence) #把文本中匹配字符替换成空字符
    new_sentence=''.join(line.split()) #去除空白
    return new_sentence
```

其中，sentence为需要进行转换的句子。

文本仅保留汉字

在处理文本时，我们通常都是仅针对文字，而符号、数字等都是没有意义的。因而可以仅保留汉字，而去除其他的语言和符号。

```
import re
def clear_character(sentence):
    #只保留中文（[\u4e00-\u9fa5]表示匹配汉字，[^\u4e00-\u9fa5]表示匹配除汉字以外的所有字符。）
    pattern = re.compile("[^\u4e00-\u9fa5]")
    line=re.sub(pattern,'',sentence) #把文本中匹配字符替换成空字符
    new_sentence=''.join(line.split()) #去除空白
    return new_sentence
```

字频统计

字频统计是数字人文研究最常用的研究方法，在实践过程中，面对大规模简体与繁体文本数据，必须借助计算机自动处理技术才能够准确高效的实现字频统计。

◆一元字频统计

一元字频统计即是对指定文本或语料库中出现的单个汉字进行频次统计。

字频统计

将待统计文本存入txt文件中，并命名为data_ch2.txt，在data_ch2.txt所在目录下新建Python文件并写入代码运行

```
import collections
# 从data_ch2.txt中读取文本
with open(r'data_ch2.txt', 'r', encoding='utf-8') as f:
    lines = f.readlines()
# 数据清洗，删去标点符号
text = ''
for line in lines:
    text += line.replace(', ', '').replace('。', '').strip()
# 统计一元字频，按频次降序排列
uchar_freq = dict(collections.Counter(text))
results = sorted(uchar_freq.items(), key=lambda x: x[1],
reverse=True)
for index,(char, freq) in enumerate(results):
    print('\t'.join([str(index+1), char, str(freq)]))
```

字频统计

◆多元字频统计

```
import collections
# 从data_ch2.txt中读取文本
with open(r'data_ch2.txt', 'r', encoding='utf-8') as f:
    lines = f.readlines()
# 数据清洗，删去标点符号
text = ''
for line in lines:
    text += line.replace(', ', '').replace('。', '').strip()
# 统计一元字频，按频次降序排列
uchar_freq = dict(collections.Counter(text))
results = sorted(uchar_freq.items(), key=lambda x: x[1],
reverse=True)
for index,(char, freq) in enumerate(results):
    print('\t'.join([str(index+1), char, str(freq)]))
```

南京農業大學
NANJING AGRICULTURAL UNIVERSITY

篆体字自动识别

文字识别技术和篆体字简介

◆传统的纸本文献数量极大且具有较高的价值，人们需要将其进行数字化，并进行存储，处理和分析研究。然而若简单将纸本文献以图片的形式存储，则难以对其内容进行分析研究。目前已有学术界和业界都聚焦于OCR（Optical Character Recognition，光学符号识别）技术，利用该技术将传统的纸本文献资料转变为计算机可以识别和存储的文字型数据，从而便于进行分析研究和处理。

◆篆体字就是俗称的篆书，狭义上包含了“大篆”和“小篆”。大篆一般指先秦时期的金文、六国文字等，小篆也叫“秦篆”，是秦始皇“书同文”的主要文字。小篆是秦汉时期主要的文字形式，直到西汉末年才逐渐被隶书取代，是秦汉时期典籍数字化和信息处理的重要对象。

TrueType字体转图片

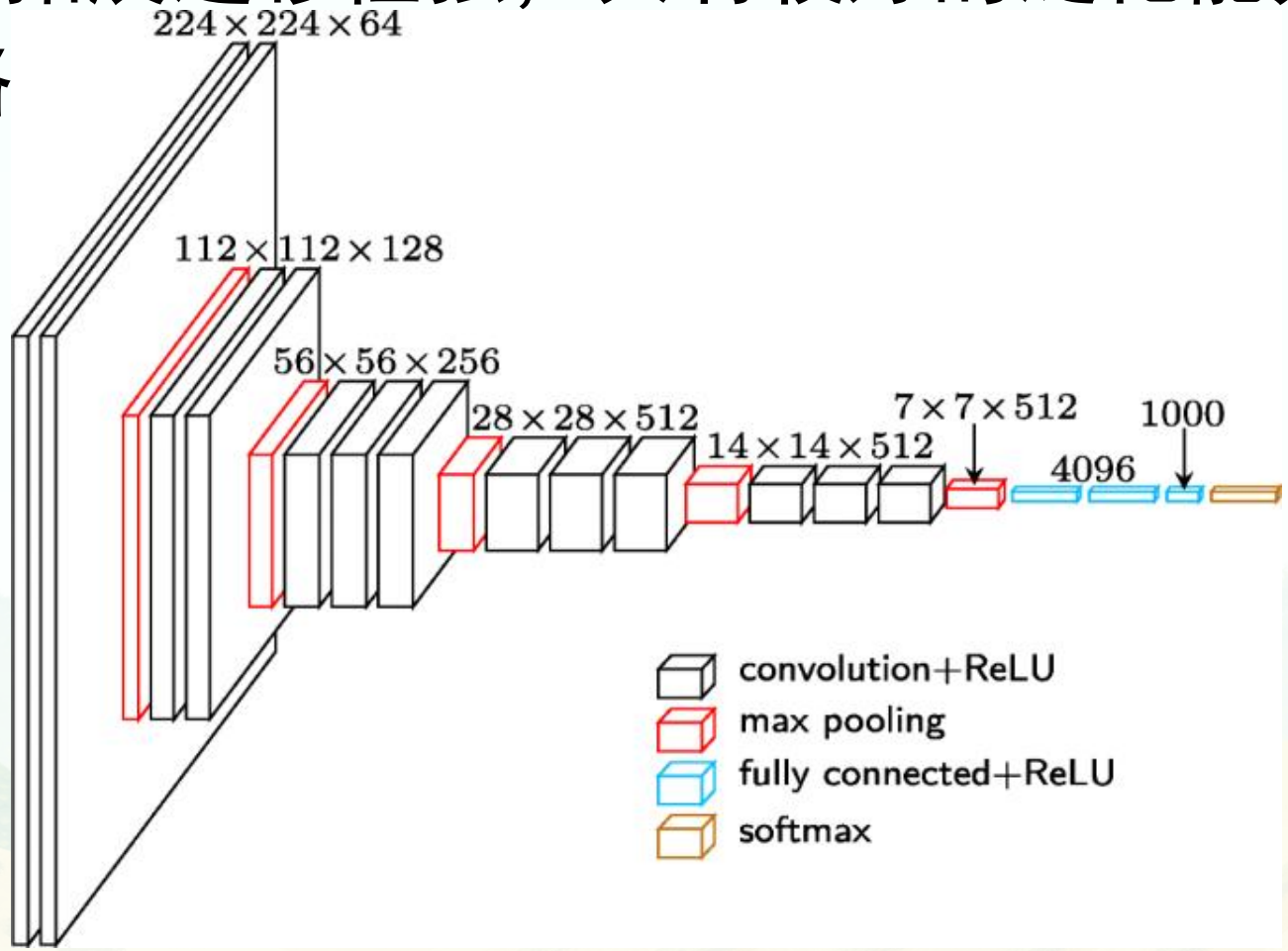
目前暂无专门面向篆体字识别任务的有标记分类数据集，通过转换篆体TrueType字体可以较为轻松地获取带有标记的篆体字图像数据，从而自行构建篆体字识别数据集。

从互联网下载了30种篆体字体，相关代码可在GitHub对应仓库（https://github.com/hsc748NLP/code-for-digital-humanities-tutorial）中获取。

通过运行TFF2IMG.py（见GitHub仓库）程序，即可实现字体到图片的转换。

VGG-16分类模型

VGG-16卷积神经网络模型是一种高效的图像分类模型，该模型结构简单、拓展迁移性强，具有较好的泛化能力。本项目的VGG-16网络其模型架构如下：



VGG-16分类模型

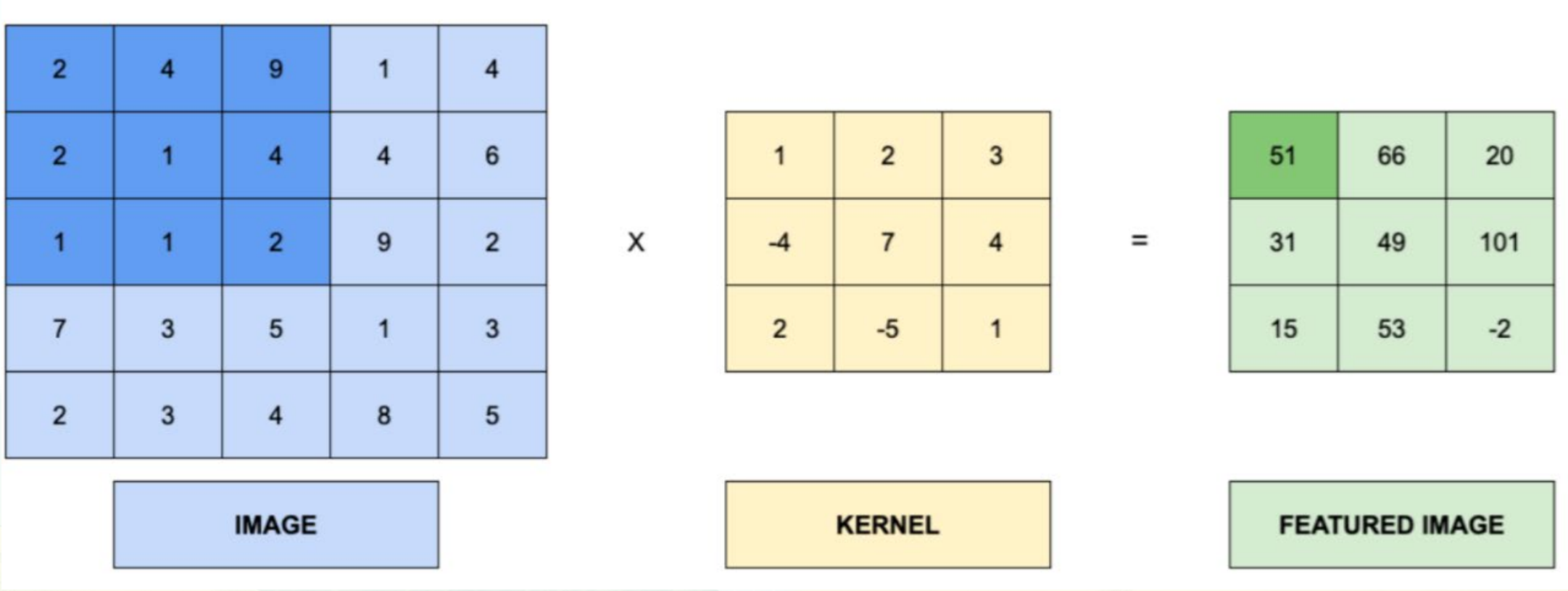
下面介绍VGG-16模型中较重要的相关概念，VGG-16模型源代码可见于GitHub仓库第一章中的vgg16.py。

◆卷积层

卷积（convolution）是一种数学运算，图像的一个子部分与一个内核（卷积核）进行卷积运算，得到一个特征图像，作为该图像的特征的代表；特征图像能够提取出图片中的特征，帮助模型进行识别。

VGG-16分类模型

卷积计算过程：



VGG-16分类模型

蓝色：图片（在当前任务中可以理解为篆体字的图片）

黄色：3×3的卷积核

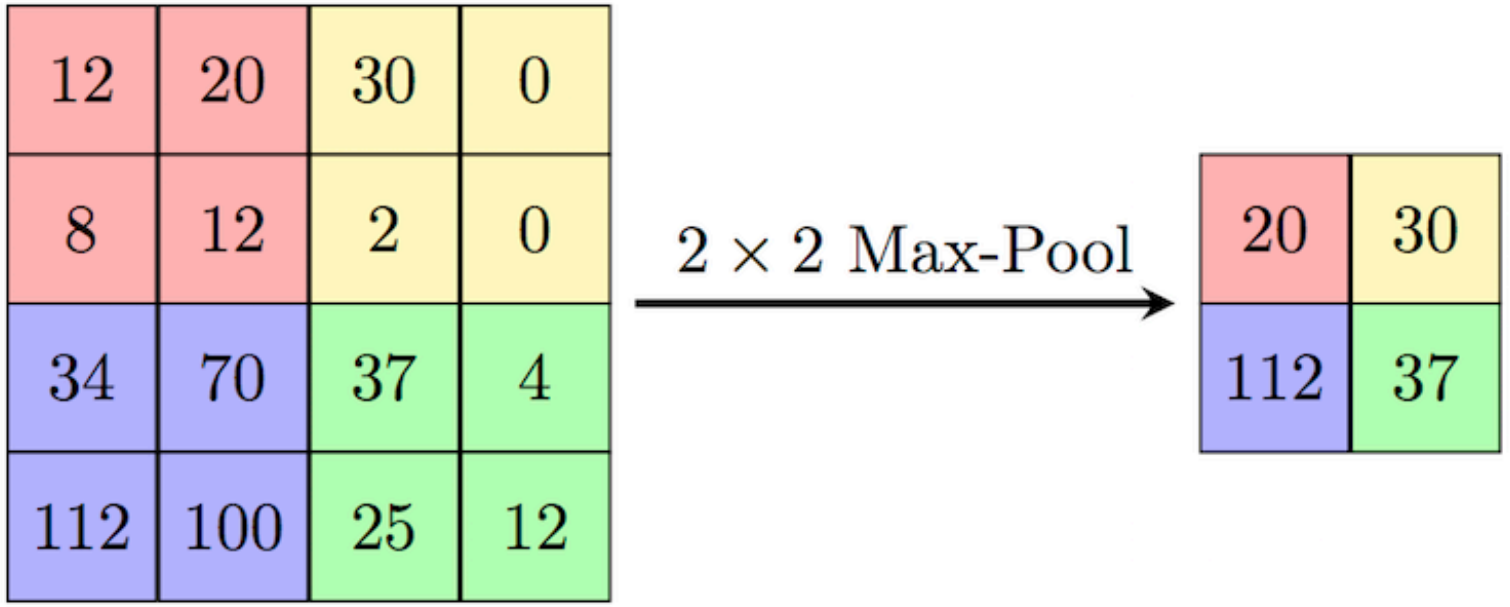
绿色：生成的特征图片

运算过程：将黄色的卷积核（kernel）在与蓝色的图片（Image）中深色的部分进行卷积计算（也就是对应的位置相乘后相加），得到了绿色特征图中深色部分的数据（feature）；随后图像上用于计算的部分进行一定方向的移动（如：向右移动一个像素-也称为strides）再次进行卷积计算，最后由此得到最终的卷积结果（绿色特征图像）。

VGG-16分类模型

◆最大池化层

在一个图片所划分出来的各个字部分中，选择最大的值作为最终的输出。如下图所示：



注意，池化的操作对象一般为卷积层输出的特征图片。

篆体字自动识别

◆模型构建

◆模型构建主要可以分为五个部分：

●设置超参数

●构建预训练模型

●设置微调部分

●添加顶层分类器与输入层

●配置模型

该部分代码见GitHub仓库中VGG16-train.py文件的第一部分

篆体字自动识别

◆数据读取

确保每个类别的图片存储在以该篆体字对应文字命名的文件夹内，指定数据集的路径。

◆数据增强

数据集类别数量众多，数据集中每个类别的图片数量相对较少。对图片进行随机旋转、平移、缩放、裁剪等转换，从而人为扩充训练集的规模。

该部分代码见GitHub仓库中VGG16-train.py文件的第二部分

篆体字自动识别

◆模型训练

```
# 当loss变化量小于指定阈值，提前结束训练
callback = tf.keras.callbacks.EarlyStopping（monitor='loss',
min_delta=0.005, patience=10）
# 训练模型
history = model.fit
（train_generator,validation_data=val_generator,epochs=EPOCHS,
callbacks= [callback]）
# 保存模型
model.save（save_path）
```

train_generator和val_generator为经过数据增强的迭代生成器，callbacks用于提前结束训练，防止过拟合。

篆体字自动识别

待模型训练结束，即可在模型保存路径看到训练好的模型文件。文件夹中包含一个pb文件，一个assets文件夹，一个变量文件夹。若已经训练完模型，要使用该模型，进行相应的任务，可以使用如下代码：

```
# 此处load_model的“（ ）”中为模型文件夹的路径
new_model = tf.keras.models.load_model（'saved_model/my_model'）
#加载测试集
Pred = new_model.predict（val_generator, verbose=1）
# 直接使用该模型进行预测
predicted_class_indices = np.argmax（Pred, axis=1）
print（predicted_class_indices）
```