Python编程与人工智能实践

应用篇：基于tflite的图像分类（MoblieNet）

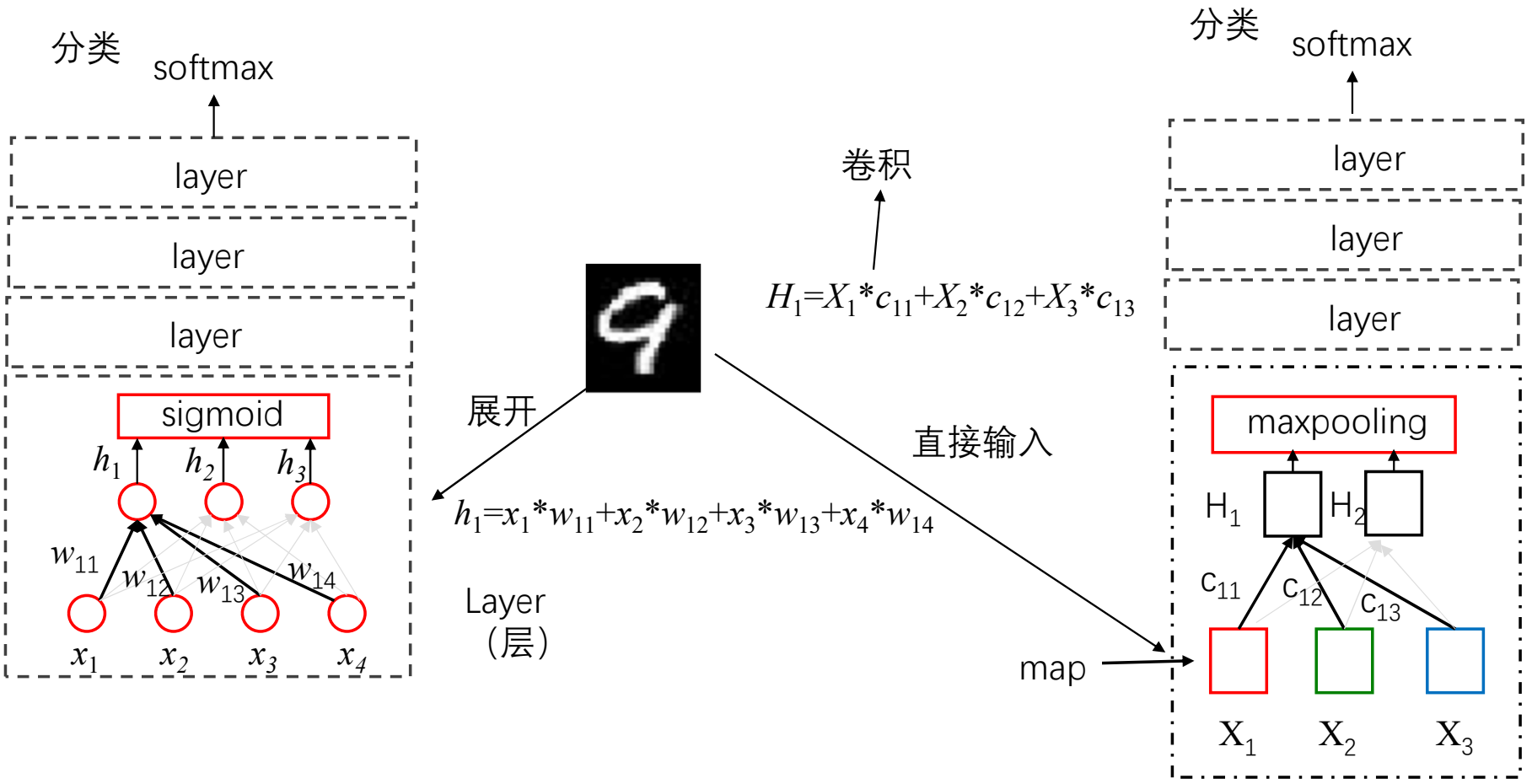
于泓
鲁东大学
信息与电气工程学院
2021.4.11

图像识别

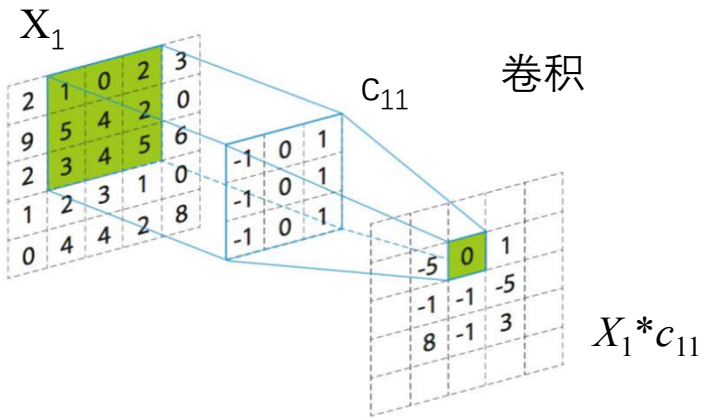
图像识别，是指利用计算机对图像进行处理、分析和理解，以识别各种不同模式的目标和对象的技术，是应用深度学习算法的一种实践应用。

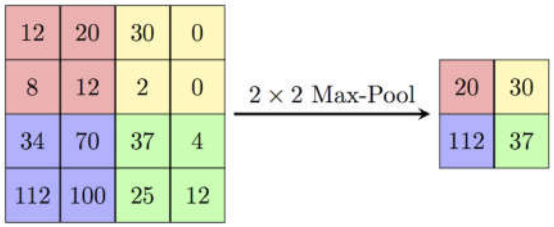
现阶段图像识别技术一般分为人脸识别与商品识别，人脸识别主要运用在安全检查、身份核验与移动支付中；商品识别主要运用在商品流通过程中，特别是无人货架、智能零售柜等无人零售领域

DNN(Deep Neural Network)与CNN(Convolutional Neural Network)



DNN(Deep Neural Network)与CNN(Convolutional Neural Network)

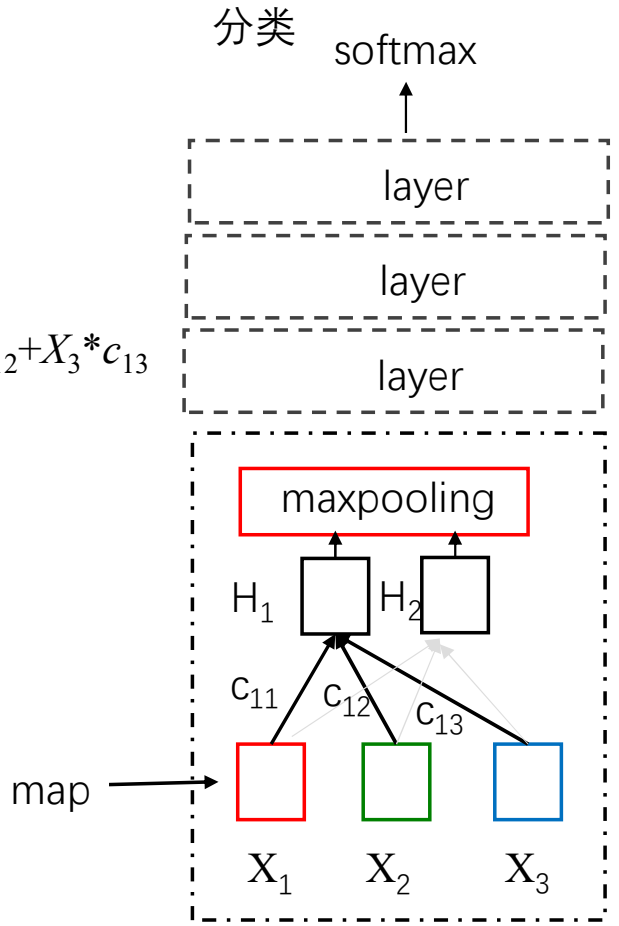




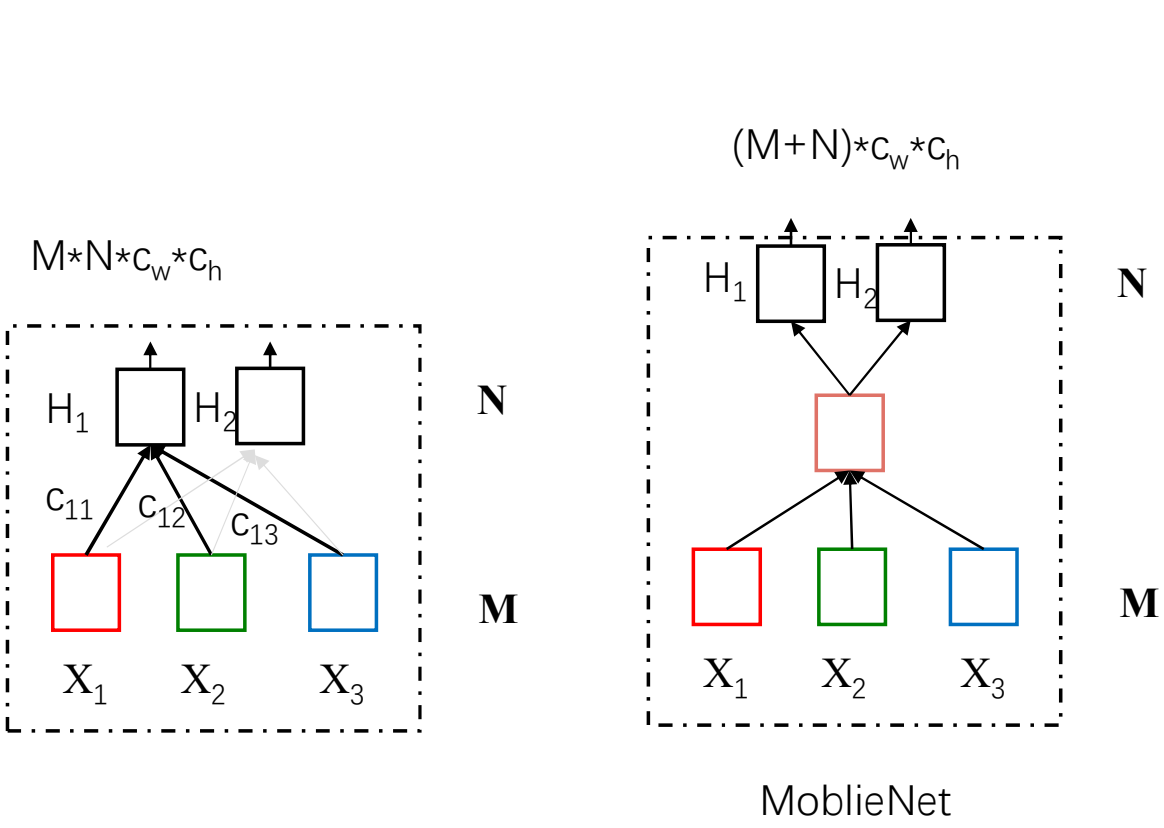
maxpooling

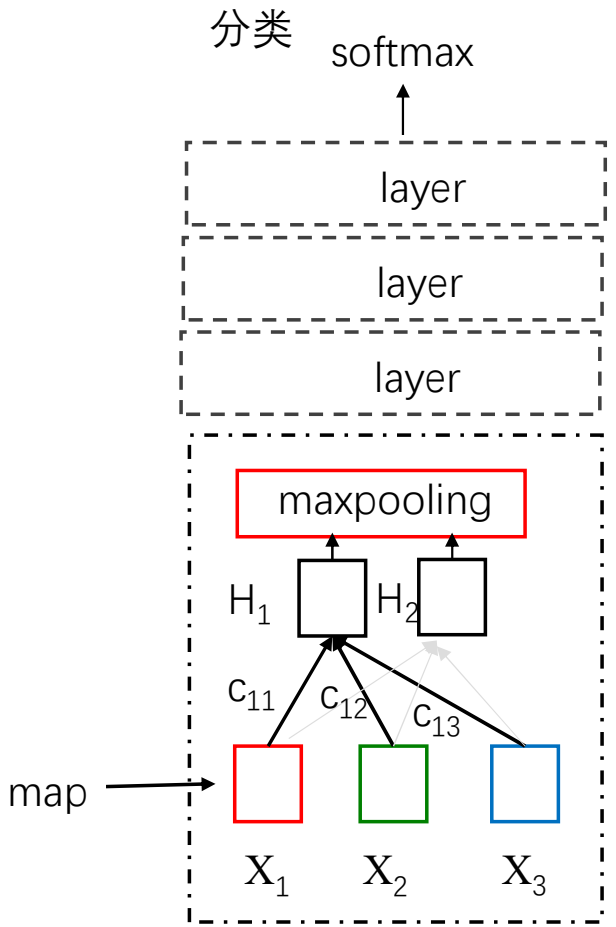
卷积

$H_1=X_1*c_{11}+X_2*c_{12}+X_3*c_{13}$



DNN(Deep Neural Network)与CNN(Convolutional Neural Network)





Tensorflow 与Tensorflow Lite

近年来随着对深度学习模型研究的不断深入以及大规模训练数据的引入在图像分类与检测的任务上取得了重大的进步。随着网络深度以及复杂度的不断增加，图像识别的精度不断提高，但是计算的复杂度也随之增加。为了能够在手机以树莓派这种运算速度较慢的手持设备上进行基于深度神经网路的图像识别任务，谷歌公式发布了针对嵌入式移动设备的深度神经网络解决方案 **TensorFlow Lite**。

安装： 下载安装包 https://www.tensorflow.org/lite/guide/python

Windows 10	3.5	https://dl.google.com/coral/python/tflite_runtime-2.1.0.post1-cp35-cp35m-win_amd64.whl
	3.6	https://dl.google.com/coral/python/tflite_runtime-2.1.0.post1-cp36-cp36m-win_amd64.whl
	3.7	https://dl.google.com/coral/python/tflite_runtime-2.1.0.post1-cp37-cp37m-win_amd64.whl

pip install tflite_runtime-2.5.0-cp38-cp38-win_amd64.whl

利用 tflite 实现图像分类任务

下载预训练模型：https://storage.googleapis.com/download.tensorflow.org/models/tflite/mobilenet_v1_1.0_224_quant_and_labels.zip

下载标签文件：https://www.tensorflow.org/lite/performance/benchmarks

labels_mobilenet_quant_v1_224.txt	2020/8/14 16:22	文本文档	11 KB
labels_mobilenet_quant_v1_224_cn_baidu.txt	2020/9/28 9:08	文本文档	11 KB
mobilenet_v1_1.0_224_quant.tflite	2020/8/14 16:22	TFLITE 文件	4,177 KB

labels_mobilenet_quant_v1_224.txt

```
1  background
2  tench
3  goldfish
4  great white shark
5  tiger shark
6  hammerhead
7  electric ray
8  stingray
9  cock
10 hen
11 ostrich
12 brambling
13 goldfinch
14 house finch
15 junco
16 indigo bunting
17 robin
18 bulbul
19 jay
20 magpie
21 chickadee
```

labels_mobilenet_quant_v1_224_cn_baidu.txt

```
1  背景
2  鲤鱼
3  金鱼
4  大白鲨
5  虎鲨
6  锤头
7  电射线
8  黄貂鱼
9  公鸡
10 母鸡
11 鸵鸟
12 吹牛
13 金翅雀
14 家朱雀
15 灯芯草雀
16 靛蓝彩旗
17 知更鸟
18 夜莺
19 松鸦
20 喜鹊
21 山雀
```

代码实现：

```
import numpy as np
import cv2
import tflite_runtime.interpreter as tflite
from PIL import Image,ImageFont,ImageDraw

def paint_chinese_opencv(im,chinese,pos,color):
    img_PIL = Image.fromarray(cv2.cvtColor(im,cv2.COLOR_BGR2RGB))
    font = ImageFont.truetype('NotoSansCJK-Bold.ttc',25,encoding="utf-8")
    fillColor = color #(255,0,0)
    position = pos #(100,100)
    # if not isinstance(chinese,unicode):
        # chinese = chinese.decode('utf-8')
    draw = ImageDraw.Draw(img_PIL)
    draw.text(position,chinese,fillColor,font)

    img = cv2.cvtColor(np.asarray(img_PIL),cv2.COLOR_RGB2BGR)
    return img
```

导入tflite

显示中文标签

```
Info of input
[{'name': 'input', 'index': 88, 'shape': array([  1, 224, 224,   3]), 'shape_signature': array([  1, 224, 224,   3]),
 'dtype': <class 'numpy.uint8'>, 'quantization': (0.0078125, 128), 'quantization_parameters': {'scales': array([0.007
8125], dtype=float32), 'zero_points': array([128]), 'quantized_dimension': 0}, 'sparsity_parameters': {}}]
Info of output
[{'name': 'MobilenetV1/Predictions/Reshape_1', 'index': 87, 'shape': array([   1, 1001]), 'shape_signature': array([
  1, 1001]), 'dtype': <class 'numpy.uint8'>, 'quantization': (0.00390625, 0), 'quantization_parameters': {'scales': a
rray([0.00390625], dtype=float32), 'zero_points': array([0]), 'quantized_dimension': 0}, 'sparsity_parameters': {}}]
```

```python
if __name__ == "__main__":
    # 输出概率最大的三个分类结果
    Top_K = 3

    # 分类模型
    file_model = "mobileNet_V1\\mobilenet_v1_1.0_224_quant.tflite"

    # 标签列表
    file_label = "mobileNet_V1\\labels_mobilenet_quant_v1_224_cn_baidu.txt"

    # 读取标签
    with open(file_label, 'r',encoding='utf-8') as f:
        labels = [line.strip() for line in f.readlines()]

    # 加载分类模型
    interpreter = tflite.Interpreter(model_path=file_model)
    interpreter.allocate_tensors()

    # 读取输入数据细节
    input_details = interpreter.get_input_details()
    print('Info of input')
    print(input_details)

    # 读取输出数据的细节
    output_details = interpreter.get_output_details()
    print('Info of output')
    print(output_details)

    # 获取输入图像的尺寸要求
    height = input_details[0]['shape'][1]
    width = input_details[0]['shape'][2]
```

```
Info of input
[{'name': 'input', 'index': 88, 'shape': array([  1, 224, 224,   3]), 'shape_signature': array([  1, 224, 224,   3]),
 'dtype': <class 'numpy.uint8'>, 'quantization': (0.0078125, 128), 'quantization_parameters': {'scales': array([0.007
8125], dtype=float32), 'zero_points': array([128]), 'quantized_dimension': 0}, 'sparsity_parameters': {}}]
Info of output
[{'name': 'MobilenetV1/Predictions/Reshape_1', 'index': 87, 'shape': array([   1, 1001]), 'shape_signature': array([
  1, 1001]), 'dtype': <class 'numpy.uint8'>, 'quantization': (0.00390625, 0), 'quantization_parameters': {'scales': a
rray([0.00390625], dtype=float32), 'zero_points': array([0]), 'quantized_dimension': 0}, 'sparsity_parameters': {}}]
```

```python
# 打开摄像头
url = "http://admin:admin@192.168.43.1:8081"
cap = cv2.VideoCapture(url)

# 初始化帧率计算
frame_rate_calc = 1
freq = cv2.getTickFrequency()

while True:

    # 获取起始时间
    t1 = cv2.getTickCount()

    # 读取一帧图像
    success, img = cap.read()

    # 获取它的尺寸
    imH,imW,_ = np.shape(img)

    # BGR 转RGB
    img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

    # 尺寸缩放适应网络输入要求
    img_resized = cv2.resize(img_rgb, (width, height))

    # 维度扩张适应网络输入要求
    input_data = np.expand_dims(img_resized, axis=0)
```

```
Info of input
[{'name': 'input', 'index': 88, 'shape': array([  1, 224, 224,   3]), 'shape_signature': array([  1, 224, 224,   3]),
 'dtype': <class 'numpy.uint8'>, 'quantization': (0.0078125, 128), 'quantization_parameters': {'scales': array([0.007
8125], dtype=float32), 'zero_points': array([128]), 'quantized_dimension': 0}, 'sparsity_parameters': {}}]
Info of output
[{'name': 'MobilenetV1/Predictions/Reshape_1', 'index': 87, 'shape': array([   1, 1001]), 'shape_signature': array([
  1, 1001]), 'dtype': <class 'numpy.uint8'>, 'quantization': (0.00390625, 0), 'quantization_parameters': {'scales': a
rray([0.00390625], dtype=float32), 'zero_points': array([0]), 'quantized_dimension': 0}, 'sparsity_parameters': {}}]
```

```python
# 数据输入网络
interpreter.set_tensor(input_details[0]['index'],input_data)

# 进行识别
interpreter.invoke()

# 获得输出
outputs = interpreter.get_tensor(output_details[0]['index'])[0]
output = np.squeeze(outputs)

# 根据量化情况对输出进行还原
if output_details[0]['dtype'] == np.uint8:
    scale, zero_point = output_details[0]['quantization']
    output = scale * (output - zero_point)

# 找到Top-K 个最大值
ordered = np.argpartition(-output, Top_K-1)

# 输出标签以及分类的概率输出
for i in range(Top_K):
    str_info = "%s %.2f%%"%(labels[ordered[i]],output[ordered[i]]*100)
    pos = (1,1+i*25)
    img = paint_chinese_opencv(img,str_info,pos,(255,0,0))

cv2.putText(img,'FPS: %.2f'%(frame_rate_calc),(imW-200,imH-20),
cv2.FONT_HERSHEY_SIMPLEX,1,(255,255,0),2,cv2.LINE_AA
```

```
    # 显示结果
    cv2.imshow('Result', img)

    # 计算帧率
    t2 = cv2.getTickCount()
    time1 = (t2-t1)/freq
    frame_rate_calc= 1/time1


    # 按q退出
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
```