**Note**

**bilibili:【计算机视觉】新手直接上手的项目实战!基于TensorFlow+CNN实现猫狗识别**

**链接：<https://www.bilibili.com/video/BV1nw411c7WT/>**

1. import warnings、warnings.filterwarnings ("ignore")：忽略警告消息，显示最新警告
2. from tensorflow.keras.preprocessing.image import ImageDataGenerator、ImageDataGenerator(featurewise\_center=False, samplewise\_center=False, featurewise\_std\_normalization=False, samplewise\_std\_normalization=False, zca\_whitening=False, zca\_epsilon=1e-06, rotation\_range=0.0, width\_shift\_range=0.0, height\_shift\_range=0.0, brightness\_range=None, shear\_range=0.0, zoom\_range=0.0, channel\_shift\_range=0.0, fill\_mode='nearest', cval=0.0, horizontal\_flip=False, vertical\_flip=False, rescale=None, preprocessing\_function=None, data\_format=None, validation\_split=0.0)：归一化图像、数据增强等图像预处理，并不增加多少倍数据，而是在每次迭代时对原图片进行一次变换，变换前后的总数据量相等。比如遍历20个epoch，原来一张图片可能经过20种变换，加强数据，使得模型更加robust。
3. featurewise\_center：布尔值，使输入数据集去中心化（均值为0），逐特征进行
4. samplewise\_center：布尔值，使输入数据的每个样本均值为0
5. featurewise\_std\_normalization：布尔值，将输入除以数据集的标准差以完成标准化，按feature执行
6. samplewise\_std\_normalization：布尔值，将输入的每个样本除以其自身的标准差
7. zca\_whitening：布尔值，对输入数据施加ZCA白化
8. zca\_epsilon：使用的eposilon，默认1e-6
9. rotation\_range：整数，图片随机转动的角度范围
10. width\_shift\_range：浮点数，一维数组或整数，图片宽度的某个比例，数据提升时图片水平偏移的幅度。float如果<1，则除以总宽度的值，如果>=1，则为宽度像素值一维数组中的随机元素。整型来自间隔(-width\_shift\_range,width\_shift\_range)间的整数个像素。width\_shift\_range=2可能值是整数[-1,0,1]，与width\_shift\_range=[-1,0,1]相同。当width\_shfit\_range=1.0时，可能值是半开区间[-1.0,1.0]之间的浮点数
11. height\_shift\_range：浮点数，图片高度的某个比例，数据提升时图片竖直偏移的幅度。具体含义与width\_shift\_range相同。
12. brightness\_range：两个float组成的元组或列表，选择亮度值的范围
13. shear\_range：浮点数，剪切强度（逆时针方向的剪切变换角度）
14. zoom\_range：浮点数或[lower, upper]，随机缩放范围，如果是浮点数，[lower, upper] = [1-zoom\_range, 1+zoom\_range]
15. channel\_shift\_range：浮点数，随机通道转换的范围
16. fill\_mode：constant、nearest、reflect或wrap，默认nearest，输入边界以外的点根据给定的模式填充。'constant': kkkkkkkk|abcd|kkkkkkkk (cval=k)，'nearest': aaaaaaaa|abcd|dddddddd，'reflect': abcddcba|abcd|dcbaabcd，'wrap': abcdabcd|abcd|abcdabcd
17. cval：浮点数或整数。用于边界之外的点的值，当fill\_mode = "constant"时
18. horizontal\_flip：布尔值，随机水平翻转。
19. vertical\_flip：布尔值，随机垂直翻转。
20. rescale：重缩放因子，默认None。如果是 None 或 0，不进行缩放，否则将数据乘以所提供的值
21. preprocessing\_function：应用于每个输入，在图像被resize和增强之后运行。该函数接收一个参数，一张图像(秩为3的numpy tensor)，输出相同shape的Numpy tensor
22. data\_format：图像数据格式，{"channels\_first", "channels\_last"} 之一。"channels\_last" 模式表示图像输入尺寸应该为(samples, height, width, channels)，"channels\_first" 模式表示输入尺寸应该为(samples, channels, height, width)。默认在Keras配置文件~/.keras/keras.json中的image\_data\_format值，从未设置是"channels\_last"
23. validation\_split：浮点型，保留用于验证集的图像比例（严格在0~1之间）
24. dtype：生成数组使用的数据类型
25. flow\_from\_directory(directory): 以文件夹路径为参数，生成经过数据提升/归一化后的数据，在一个无限循环中无限产生batch数据
26. directory：目标文件夹路径，对于每一个类，该文件夹都要包含一个子文件夹，子文件夹中任何JPG、PNG、BNP、PPM的图片都会被生成器使用
27. target\_size：整数tuple，默认为(256, 256)，图像将被resize成该尺寸
28. color\_mode：颜色模式，默认为"rgb"，可以为"grayscale"，代表是否会被转换为单通道或三通道的图片
29. classes：可选参数，为子文件夹的列表，默认为None，从directory下的子文件夹名称/结构自动推断。每一个子文件夹都会被认为是一个新的类，类别的顺序将按照字母表顺序映射到标签值。通过属性class\_indices可获得文件夹名与类的序号的对应字典
30. class\_mode："categorical", "binary", "sparse"或None之一，默认"categorical。决定返回的标签数组的形式， "categorical"返回2D的one-hot编码标签，"binary"返回1D的二值标签，"sparse"返回1D的整数标签，None则不返回任何标签，仅生成batch数据，在使用model.predict\_generator()和model.evaluate\_generator()等函数时会用到
31. batch\_size：batch数据的大小，默认32
32. shuffle：是否打乱数据，默认为True
33. seed：可选参数，打乱数据和进行变换时的随机数种子
34. save\_to\_dir：None或字符串，该参数能让你将提升后的图片保存起来，用以可视化
35. save\_prefix：字符串，保存提升后图片时使用的前缀，仅当设置save\_to\_dir时生效
36. save\_format："png"或"jpeg"之一，指定保存图片的数据格式，默认"jpeg"
37. flollow\_links：是否访问子文件夹中的软链接
38. model.fit\_generator(self, generator, steps\_per\_epoch, epochs=1, verbose=1, callbacks=None, validation\_data=None, validation\_steps=None, class\_weight=None, max\_q\_size=10, workers=1, pickle\_safe=False, initial\_epoch=0)：利用生成器，分批次向模型送入数据的方式，可以有效节省单次内存的消耗，在之后被弃用，用model.fit()代替
39. generator：一般是一个生成器函数
40. steps\_per\_epochs：在每个epoch中生成器执行生成数据的次数
41. epochs：训练过程中需要迭代的次数
42. verbose：默认为1，指在训练过程中日志的显示模式，取 1 时表示“进度条模式”，取2时表示“每轮一行”，取0时表示“安静模式”
43. validation\_data、validation\_steps：验证集情况，使用和generator, steps\_per\_epoch相同
44. models.fit\_generator()返回一个history对象，？.history：记录训练过程中，训练集损失和评估值、验证集损失和评估值
45. model.fit(x=None, y=None, batch\_size=None, epochs=1, verbose=1, callbacks=None, validation\_split=0.0, validation\_data=None, shuffle=True, class\_weight=None, sample\_weight=None, initial\_epoch=0, steps\_per\_epoch=None, validation\_steps=None)：在使用fit函数的时候，需要有batch\_size，但是在使用fit\_generator时需要有steps\_per\_epoch
46. x: 训练数据的 Numpy 数组（如果模型只有一个输入）， 或者是 Numpy 数组的列表（如果模型有多个输入）。 如果模型中的输入层被命名，你也可以传递一个字典，将输入层名称映射到 Numpy 数组。 如果从本地框架张量馈送（例如 TensorFlow 数据张量）数据，x 可以是 None（默认）。
47. y: 目标（标签）数据的 Numpy 数组（如果模型只有一个输出）， 或者是 Numpy 数组的列表（如果模型有多个输出）。 如果模型中的输出层被命名，你也可以传递一个字典，将输出层名称映射到 Numpy 数组。 如果从本地框架张量馈送（例如 TensorFlow 数据张量）数据，y 可以是 None（默认）。
48. batch\_size: 整数或 None。每次梯度更新的样本数。如果未指定，默认为 32。
49. epochs: 整数。训练模型迭代轮次。一个轮次是在整个 x 和 y 上的一轮迭代。 请注意，与 initial\_epoch 一起，epochs 被理解为 「最终轮次」。模型并不是训练了 epochs 轮，而是到第 epochs 轮停止训练。
50. verbose: 0, 1 或 2。日志显示模式。0 =安静模式, 1 =进度条, 2 =每轮一行。
51. callbacks: 一系列的 keras.callbacks.Callback 实例。一系列可以在训练时使用的回调函数。 详见 callbacks。
52. validation\_split: 0 和 1 之间的浮点数。用作验证集的训练数据的比例。 模型将分出一部分不会被训练的验证数据，并将在每一轮结束时评估这些验证数据的误差和任何其他模型指标。 验证数据是混洗之前 x 和y 数据的最后一部分样本中。
53. validation\_data: 元组 (x\_val，y\_val) 或元组 (x\_val，y\_val，val\_sample\_weights)， 用来评估损失，以及在每轮结束时的任何模型度量指标。 模型将不会在这个数据上进行训练。这个参数会覆盖 validation\_split。
54. shuffle: 布尔值（是否在每轮迭代之前混洗数据）或者 字符串 (batch)。 batch 是处理 HDF5 数据限制的特殊选项，它对一个 batch 内部的数据进行混洗。 当 steps\_per\_epoch 非 None 时，这个参数无效。
55. class\_weight: 可选的字典，用来映射类索引（整数）到权重（浮点）值，用于加权损失函数（仅在训练期间）。 这可能有助于告诉模型 「更多关注」来自代表性不足的类的样本。
56. sample\_weight: 训练样本的可选 Numpy 权重数组，用于对损失函数进行加权（仅在训练期间）。 您可以传递与输入样本长度相同的平坦（1D）Numpy 数组（权重和样本之间的 1:1 映射）， 或者在时序数据的情况下，可以传递尺寸为 (samples, sequence\_length) 的 2D 数组，以对每个样本的每个时间步施加不同的权重。 在这种情况下，你应该确保在 compile() 中指定 sample\_weight\_mode="temporal"。
57. initial\_epoch: 整数。开始训练的轮次（有助于恢复之前的训练）。
58. steps\_per\_epoch: 整数或 None。 在声明一个轮次完成并开始下一个轮次之前的总步数（样品批次）。 使用 TensorFlow 数据张量等输入张量进行训练时，默认值 None 等于数据集中样本的数量除以 batch 的大小，如果无法确定，则为 1。
59. validation\_steps: 只有在指定了 steps\_per\_epoch 时才有用。停止前要验证的总步数（批次样本）
60. model = tf.keras.models.Sequential([网络结构])：描述各层网络
61. tf.keras.layers.Conv2D(filters, kernel\_size, strides=(1, 1), padding='valid', data\_format=None, dilation\_rate=(1, 1), activation=None)：
62. filter：卷积核的个数
63. kenel\_size：卷积核尺寸，正方形用一个整数表示，长方形需要明确指明高用h表示，宽用w表示，可以用元组或者列表的形式表示两者(如：[h,w]或者{h, w})
64. strides：滑动步长，默认为1(1,1)，也可以设置其他步长(纵向步长，横向步长)
65. padding：补零策略，当padding = 'same' 时，全零填充，当padding = 'valid’，则不需要填充，不区分大小写
66. data\_format：输入数据的格式，值有两种channels\_first(输入和输出的shape为(batch\_size, channels, height, width)，即为(图片数量，通道数，长，宽))、channels\_last(默认值通道数为左最后一个)
67. dalition\_rate：数组或列表，卷积核膨胀系数(将卷积核进行形状膨胀，新位置用0填充)。新卷积核的尺寸核膨胀系数的计算：原卷积核的尺寸S，膨胀系数K，则膨胀后的卷积核尺寸为size = K\*(S-1)+1
68. activaton：激活函数，相当于经过卷积输出后，再经过一次激活函数(常见的激活函数有relu，softmax，selu)
69. keras.layers.MaxPooling2D(pool\_size=(2, 2), strides=None, padding='valid', data\_format=None)：
70. pool\_size：池化窗口大小
71. strides：池化步长，默认值等于 pool\_size
72. padding： 'VALID' 或 'SAME'，'VALID'表示无填充，'SAME'表示用0填充
73. data\_format：表示输入张量的维度顺序，默认为 [batch, height, width, channel]
74. tf.keras.layers.Flatten()：拉直层/扁平层，把输入特征拉直为一维数组，不含计算参数
75. tf.keras.layers.Dense(units, activation=None, use\_bias=True, kernel\_initializer='glorot\_uniform', bias\_initializer='zeros', kernel\_regularizer=None, bias\_regularizer=None, activity\_regularizer=None, kernel\_constraint=None, bias\_constraint=None, \*\*kwargs)
76. units：输出的单元个数，此处应该是int型，与输出的N为tensor中最后一位相同
77. activation：激活函数，相当于对输出的结果经过一次激活函数的运算，得到新的值，也可以不选择激活函数
78. use\_bias：布尔值，是否使用偏移向量
79. kernel\_initializer：核权重矩阵的初始值设定项
80. bias\_initializer：偏差向量的初始值设定项
81. kernel\_regularizer：正则化函数应用于核权矩阵
82. bias\_regularizer：应用于偏差向量的正则化函数
83. activity\_regularizer：应用于层输出的正则化函数(“激活”)
84. kernel\_constraint：应用于核权矩阵的约束函数
85. bias\_constraint：对偏置向量施加约束函数
86. model.summary()：输出模型结果
87. model.compile(optimizer, loss = None, metrics = None, loss\_weights = None, sample\_weight\_mode = None, weighted\_metrics = None, target\_tensors = None)：
88. optimizer：优化器，用于控制梯度裁剪，可用参数Adam/Adamax/Nadam
89. loss：损失函数（或称目标函数、优化评分函数），可用均方误差、平均绝对误差、平均绝对百分比误差、均方对数误差、squared\_hinge、hinge、categorical\_hinge、预测误差的双曲余弦的对数logcosh、分类交叉熵categorical\_crossentropy、稀疏的分类交叉熵sparse\_categorical\_crossentropy、二元交叉熵binary\_crossentropy、kullback\_leibler\_divergence、泊松poisson、余弦值cosine\_proximity
90. metrics：评价函数，当模型编译后（compile），评价函数应该作为 metrics 的参数来输入。评价函数和损失函数相似，只不过评价函数的结果不会用于训练过程中。内置有binary\_accuracy、categorical\_accuracy、sparse\_categorical\_accuracy、top\_k\_categorical\_accuracy、sparse\_top\_k\_categorical\_accuracy，也可以自定义
91. 图像库：from PIL import Image或import opencv2
92. from keras.preprocessing import image：加载和预处理图像
93. import keras.backend as K：使用抽象keras backend来写新代码实现兼容。keras是一种基于模块的高级深度学习开发框架，它并没有仅依赖于某一种高速底层张量库，而是对各种底层张量库进行高层模块封装，让底层库完成诸如张量积、卷积操作。目前，Keras提供了三种后端实现：TensorFlow后端，Theano后端和CNTK后端
94. from tensorflow.keras import layers：用于深度学习的更高层次封装的 API，方便使用
95. from tensorflow.keras import Model：导入模型块
96. from tensorflow.keras.applications.renet import ResNet101：迁移学习中，导入官网API中的ResNet101网络
97. 回调函数callbacks：控制正在训练的模型，在验证集准确率不再上升时终止训练。存储模型的状态，采取措施打断训练，保存模型，加载不同的权重或替代模型状态。可用于模型断点续训（保存当前模型的所有权重）、提早结束（当模型的损失不再下降的时候就终止训练，保存最优模型）、动态调整训练参数（如优化的学习率）。

callbacks = [

tf.keras.callbacks.EarlyStoppong(patience = 2, monitor = 'val\_loss'), # 如果连续两个epoch还没降低就停止

tf.keras.callbacks.LearningRateScheduler # 动态改变学习率

tf.keras.callbacks.ModelCheckpoint # 保存模型

tf.keras.callbacks.Callback] # 自定义方法

1. TFRecords作用：高效读取数据，将数据进行序列化存储，便于网络流式读取数据，是常用的存储二进制序列的方法。tf.example类将数据表示为{"string": value}形式的meassage类型，常用于写入、读取TFRecords数据。tf.example格式有：tf.train.BytesList用于string和byte，tf.train.FloatList用于float和double，tf.train.Int64List用于枚举enum、bool、int32、unit32、int64
2. tf.constant(value, dtype=None, shape=None, name='Const', verify\_shape=False)：创建常量的函数，value可以是数值或列表
3. Tensor.numpy()：将Tensor转化为ndarray，这里的Tensor可以是标量或者向量（与item()不同）转换前后的dtype不会改变
4. tf.train.Feature：有三个属性：tf.train.bytes\_list tf.train.float\_list tf.train.int64\_list，根据属性值来设置tf.train.Feature的属性
5. tf.train.Features：有属性feature，一般设置方法是传入一个字典，字典的key是字符串（feature名），而值是tf.train.Feature对象
6. tf.io.TFRecordWriter (filename, options=None)：输出TFrecord的API，其中，第二个参数用来控制文件的输出配置，一般不用管。第一个参数是要保存的文件名，调用该函数后，会返回一个Writer实例
7. tf.train.Example(features\_extern)：将features编码数据封装成特定的PB协议格式
8. example.SerializeToString()：将example数据系列化为字符串
9. import datatime：处理日期和时间
10. tf.image.decode\_jpeg：别名tf.io.decode\_jpeg，用于将JPEG编码的图像解码为uint8张量，jpg也适用
11. 为了解析每个输入样本每一列数据，需要定义一个解析字典。tensorflow提供三种方式：FixedLenFeature、VarLenFeature、FixedLenSequenceFeature，分别解析定长特征、变长特征、定长序列特征。FixedLenFeature() 函数有三个参数：输入数据的shape、输入的数据类型dtype、如果示例缺少此功能则使用该值default\_value，它必须与dtype和指定shape兼容
12. tf.io.parse\_single\_example( serialized, features, name=None, example\_names=None)：从TFRecords文件中读取数据，使用tf.parse\_single\_example()将Example协议内存块(protocol buffer)解析为张量。
13. serialized：一个标量字符串张量，一个序列化的例子
14. features：dict将特性键映射到FixedLenFeature或VarLenFeature值
15. name：此操作的名称(可选)
16. example\_names：(可选)标量字符串张量，关联的名称
17. tf.data.Dataset.shuffle(buffer\_size, seed=None, reshuffle\_each\_iteration=None, name=None)：随机洗牌此数据集的元素
18. buffer\_size：tf.int64 /tf.Tensor，表示新数据集将从此数据集中采样的元素数
19. seed：可选，tf.int64 /tf.Tensor，表示将用于创建分布的随机种子
20. reshuffle\_each\_iteration：可选，tf.bool，如果为True，则表示每次迭代数据集时都应伪随机地重新洗牌，默认为True
21. name：可选，tf.data操作的名称