PROJECT REPORT

ON

APP Icons Identifcation using Image Processing

By:

Ashish Kishor

OpenGL App Testing using Image Processing

Overview

For app testing, screen details (example: text, resource-id etc.) of elements present in an app are required. The details are then used to identify various elements on screen and perform operations on them. Screen details are gathered by uiautomator dump. But for OpenGL apps like camera, the uiautomator dump does not contain much information about the screen and thus it becomes difficult to fetch information for screen elements which results in difficult to test OpenGL apps. Convolutional Neural Network can be used to identify various text, icons present on any screen. If CNN model is trained with adequate number of images, we can appropriately predict the category of an image and thus process screen elements.

Language Used

Convolutional Neural Network

Convolutional neural networks (CNN) is a special architecture of artificial neural networks.

Instead of the image, the computer sees an array of pixels. For example, if image size is 300 x 300. In this case, the size of the array will be 300x300x3. Where 300 is width, next 300 is height and 3 is RGB channel values. The computer is assigned a value from 0 to 255 to each of these numbers. Тhis value describes the intensity of the pixel at each point.

The Convolution layer is always the first. Тhe image (matrix with pixel values) is entered into it. Next the software selects a smaller matrix there, which is called a filter (or neuron, or core).

The filter’s task is to multiply its values by the original pixel values. All these multiplications are summed up. One number is obtained in the end. Since the filter has read the image only in the upper left corner, it moves further and further right by 1 unit performing a similar operation. After passing the filter across all positions, a matrix is obtained, but smaller then a input matrix.

The most important parameter in a convolutional neuron is the filter size, let’s say we have a layer with filter size 5\*5\*3. Also, assume that the input that’s fed to convolutional neuron is an input image of size of 32\*32 with 3 channels.

When we calculate the dot product it’s a matrix multiplication of 5\*5\*3 sized chunk with 5\*5\*3 sized filter.

The nonlinear layer

ReLU stands for Rectified Linear Unit for a non-linear operation. The output is ƒ(x) = max(0,x).Why ReLU is important : ReLU’s purpose is to introduce non-linearity in our ConvNet. Since, the real world data would want our ConvNet to learn would be non-negative linear values.

There are other nonlinear functions such as tanh or sigmoid can also be used instead of ReLU. Most of the data scientists uses ReLU since performance wise ReLU is better than other two.

The pooling layer follows the nonlinear layer. It works with width and height of the image and performs a down sampling operation on them. As a result the image volume is reduced. This means that if some

features (as for example boundaries) have already been identified in the previous convolution operation, than a detailed image is no longer needed for further processing, and it is compressed to less detailed pictures.

After completion of series of convolutional, nonlinear and pooling layers, it is necessary to attach a fully connected layer. This layer takes the output information from convolutional networks. Attaching a fully connected layer to the end of the network results in an N dimensional vector, where N is the amount of classes from which the model selects the desired class.

Probabilistic Model

Multi-class classification is the problem of classifying objects into one of the more than two classes. The goal of classification is to construct a classifier which, given a new test object, will predict the class label from the set of possible k classes. In an “ordinary” classification problem, the goal is to minimize the loss function by predicting correct labels on the test set. In contrast, a probabilistic classifier outputs, for a given test object, the probability distribution over the set of k classes. Probabilistic classifiers allow to express a degree of confidence about the classification of the test object.

Understanding Categorical Cross-Entropy Loss, Binary Cross-Entropy Loss, Softmax Loss

1. Multi-Class Classification

One-of-many classification. Each sample can belong to ONE of CC classes. The CNN will have CC output neurons that can be gathered in a vectors (Scores). The target (ground truth) vector tt will be a one-hot vector with a positive class and C−1C−1 negative classes.

This task is treated as a single classification problem of samples in one of CC classes.

2. Multi-Label Classification

Each sample can belong to more than one class. The CNN will have as well CC output neurons. The target vector it can have more than a positive class, so it will be a vector of 0s and 1s with CC dimensionality.

This task is treated as CC different binary (C′=2, t′=0 or t′=1)(C′=2,t′=0 or t′=1) and independent classification problems, where each output neuron decides if a sample belongs to a class or not.

Output Activation Functions

These functions are transformations we apply to vectors coming out from CNNs (ss) before the loss computation.

1. Sigmoid

It squashes a vector in the range (0, 1). It is applied independently to each element of si. It’s also called logistic function.

2. Softmax

Softmax it’s a function, not a loss. It squashes a vector in the range (0, 1) and all the resulting elements add up to 1. It is applied to the output scores ss. As elements represent a class, they can be interpreted as class probabilities.

The Softmax function cannot be applied independently to each si , since it depends on all elements of ss.

Activation functions are used to transform vectors before computing the loss in the training phase. In testing, when the loss is no longer applied, activation functions are also used to get the CNN outputs.

Crossentrophy

Categorical Cross-Entropy loss

Also called Softmax Loss. It is a Softmax activation plus a Cross-Entropy loss. If we use this loss, we will train a CNN to output a probability over the classes for each image. It is used for multi-class classification.

Binary Cross-Entropy Loss

Also called Sigmoid Cross-Entropy loss. It is a Sigmoid activation plus a Cross-Entropy loss. Unlike Softmax loss it is independent for each vector component (class), meaning that the loss computed for every CNN output vector component is not affected by other component values. That’s why it is

Keras

Keras is an open-source neural-network library written in Python. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible. Keras contains numerous implementations of commonly used neural-network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier. The code is hosted on GitHub, and community support forums include the GitHub issues page, and a Slack channel.In addition to standard neural networks, Keras has support for convolutional and recurrent neural networks. It supports other common utility layers like dropout, batch normalization, and pooling. Keras allows users to productize deep models on smartphones (iOS and Android), on the web, or on the Java Virtual Machine. It also allows use of distributed training of deep-learning models on clusters of Graphics Processing Units (GPU) and Tensor processing units (TPU).

• Models

1. Sequential model

The core data structure of Keras is a model, a way to organize layers. The simplest type of model is the sequential model, a linear stack of layers.

Import model

from keras.models import Sequential

Create instance

model = Sequential()

2. Functional model

For more complex architectures, you should use the Keras functional API, which allows to build arbitrary graphs of layers.

• Layers

Keras layers are the fundamental building block of keras models. Layers are created using a wide variety of layer\_ functions and are typically composed together by stacking calls to them using the pipe %>% operator.

A wide variety of layers are available, including:

o Convolutional Layers

o Pooling Layers

o Activation Layers

o Dropout Layers

o Fully Connected Layers

Stacking layers is as easy as .add():

Importing Layers

from keras.layers import Dense,Covolution,Activation,Flatten

Adding Layers

classifier.add(type\_of\_layer())

classifier.add(Dense(units=64, activation='relu', input\_dim=100))

classifier.add(Dense(units=10, activation='softmax'))

Flatten

Classifier.add(Flatten())

Flattens the input. Does not affect the batch size.

Dropout

Classifier.add(Dropout(rate))

Applies Dropout to the input. Dropout consists in randomly setting a fraction rate of input units to 0 at each update during training time, which helps prevent overfitting.

Arguments

rate: float between 0 and 1. Fraction of the input units to drop.

Activation

Activations can either be used through an activation layer, or through the activation argument supported by all forward layers.

classifier.add(Dense(64, activation='softmax'))

Arguments

x: Input tensor.

axis: Integer, axis along which the softmax normalization is applied.

Returns

Tensor, output of softmax transformation.

Image Datagenerator

ImageDataGenerator has the following arguments:

1. Rescale -  The reason for normalizing the input has to do with numerical stability and convergence (technically you do not need it, but with it, the neural network has a higher chance of converging and the gradient descent/adam algorithm is way more likely to be stable)

2. Shear\_range - shear intensity, used for linear mapping that displaces each point in a fixed direction

3. Zoom\_range - use for random zooming

4. Horizontal\_flip - unlike other arguments has boolean type, used for randomly flipping inputs horizontally

Batch size the number of training examples in one forward/backward pass (or for 1 epoch, which is expected).Then the already described Image Data Generator is added for training and tasting samples. But it has a new transformation, which is called rescale. It multiplies the data by the given value. Further, the target size follows. It shows width and height to which images will be resized. Next, the batch size is added. Finally binary class mode is set. Training is possible with the help of the fit\_generator. Here it is important to indicate a number of epochs, which defines for how many times the training will repeat. 1 epoch is 1 forward pass and 1 backward pass over all the training examples. Also, in this section steps\_per\_epoch and validation\_steps are set. Steps\_per\_epoch (or number of iterations) shows total number of steps, which is used to declare one epoch finished and begin the next. Typically this number is equal to the number of samples for training (in my case it is 700: 200 photos of setting and shutter) divided by the batch size (16). It means that the number of iterations of validation steps: 200 / 16 = 12. Validation\_steps is total number of steps (batches of samples) to validate before stopping.

Matplotlib

Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib consists of several plots like line, bar, scatter, histogram etc. Matplotlib comes with a wide variety of plots. Plots helps to understand trends, patterns, and to make correlations. They’re typically instruments for reasoning about quantitative information.

To Import Matplotlib

import matplotlib

To import pyplot

import matplotlib.pyplot as plt

To plot

plt.plot()

To show the plotted graph

plt.show()

Skimage

Scikit-image is an image processing Python package that works with numpy arrays which is a collection of algorithms for image processing. Let’s discusses how to deal with images into set of information and its some application in real world.

1. Regioprops Module

Measure properties of labeled image regions.

Parameters

Label\_image : (N, M) and array

Labeled input image: Previously, label\_image was processed by numpy.squeeze and so any number of singleton dimensions was allowed. This resulted in inconsistent handling of images with singleton dimensions. To recover the old behavior, use regionprops(np.squeeze(label\_image), ...).

Intensity\_image : (N, M) and array, optional

Returns

Properties: list of Region Properties

Each item describes one labeled region, and can be accessed using the attributes.

Bboxtuple

Bounding box (min\_row, min\_col, max\_row, max\_col). Pixels belonging to the bounding box are in the half-open interval [min\_row; max\_row) and [min\_col; max\_col).

Python Imaging Library

Python Imaging Library (abbreviated as PIL) (in newer versions known as Pillow) is a free library for the Python programming language that adds support for opening, manipulating, and saving many different image file formats. It is available for Windows, Mac OS X and Linux.

Installing PIL

pip install pillow

Importing PIL

from PIL import ImageDraw

1. Image Module

Operations with Images

• Open a particular image from a path

img = Image.open(path)

• Save changes in image

img.save(path, format)

• Cropping an Image

Image.crop(box) takes a 4-tuple (left, upper, right, lower) pixel coordinate, and returns a rectangular region from the used image.

img = Image.open(path)

width, height = img.size

area = (0, 0, width/2, height/2)

img = img.crop(area)

2. ImageDraw Module

The module provide simple 2D graphics for Image objects. You can use this module to create new images, annotate or retouch existing images, and to generate graphics on the fly for web use.

• Draw a line

draw.line((0, 0) + im.size, fill=128)

fill: color to fill

size: size of the line

• Draw over image

Draw=ImageDraw.draw(img)

OpenCV

OpenCV (open source computer vision) is a very powerful library for image processing and machine learning tasks which also supports Tensorflow, Torch/Pytorch and Caffe. The library is cross platform and you can pip install it (where you are using it with Python) with CPU support. Alternatively, where for example you want to use it with GPU support, you can build it from source, which is more involved.

Install library

pip install opencv

Import library

Import cv2

Operation with image

• To read and image

image = cv2.imread(path)

• To covert image into grayscale

gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

Numpy

NumPy is the fundamental package for scientific computing with Python. It contains among other things:

• a powerful N-dimensional array object

• sophisticated (broadcasting) functions

• tools for integrating C/C++ and Fortran code

• useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

Edge Detection

Edge detection is useful for finding boundaries of objects in an image it is effective for segmentation purposes. Using the popular Canny algorithm (developed by John F. Canny in 1986), we can find the edges in the image. We provide parameters to the cv2.Canny function:

Img : The gray image.

MinVal : A minimum threshold, in our case 30 .

MaxVal : The maximum threshold which is 150 in our example.

Aperture\_size : The Sobel kernel size. By default this value is 3.Different values for the minimum and maximum thresholds will return different edge maps.

1.4 UIAutomator

UI Automator is a UI testing framework suitable for cross-app functional UI testing across system and installed apps. The UI Automator testing framework provides a set of APIs to build UI tests that perform interactions on user apps and system apps. The UI Automator APIs allows you to perform operations such as opening the app launcher in a test device. The UI Automator testing framework is well-suited for writing black box-style automated tests, where the test code does not rely on internal implementation details of the target app. The UI Automator APIs allow you to write robust tests without needing to know about the implementation details of the app that you are targeting. You can use these APIs to capture and manipulate UI components across multiple apps:

This project uses python wrapper of android UI testing framework. It works on Android 4.1+ simply with Android device attached via adb, no need to install anything on Android device.

Install UI Automator

$ pip install uiautomator

Import UI Automator

from uiautomator import device as d

Key Event Actions of the device

d.screenshot()

d.click(x,y)

d.sleep()

Implementation

1.1 Binary Classification Convolutional Neural Network Model

It started running by processing the model layer by layer it start with the very first layer convolution 2D. The number 32 shows the amount of output filter in the convolution. Numbers 3, 3 correspond to the kernel size, which determinate the width and height of the 2D convolution window. An important component of the first convolution layer is an input shape, which is the input array of pixels. The activation function of this model is relu. This function sets the zero threshold and looks like: f(x) = max (0,x). If x > 0 — the volume of the array of pixels remains the same, and if x < 0 — it cuts off unnecessary details in the channel. Max Pooling 2D layer is pooling operation for spatial data. Numbers 2, 2 denote the pool size, which halves the input in both spatial dimension. After three groups of layers there are two fully connected layers. Flatten performs the input role. Next is Dense — densely connected layer with the value of the output space (64) and relu activation function. It follows Dropout, which is preventing overfitting. Overfitting is the phenomenon when the constructed model recognizes the examples from the training sample, but works relatively poorly on the examples of the test sample. Тhe last fully connected layer has 1 output and sigmoid activation function. After adding a sufficient number of layers the model is compiled. At this moment keras communicates with tensorFlow for construction of the model. During model compilation it is important to write a loss function and an optimizer algorithm. It looks like: classifier.compile(loss= ‘name\_of\_loss\_function’, optimizer= ‘name\_of\_optimizer\_algo’ ) The loss function shows the accuracy of each prediction made by the classifier.Next step is model compiling. It has a binary crossentropy loss function, which will show the sum of all individual losses. The optimizer algorithm is adam. The accuracy metrics shows the performance of the model. Before training it is important to scale data for their further use. After model construction it is time for model training. In this phase, the model is trained using training data and expected output for this data. It’s look this way: classifier.fit(training\_data, expected\_output). Тhis class can create a number of random transformations, which helps to increase the number of images when it is needed. Also, steps\_per\_epoch and validation\_steps are set. Steps\_per\_epoch (or number of iterations) shows total number of steps, which is used to declare one epoch finished and begin the next. Typically this number is equal to the number of samples for training (in my case it is 700: 200 photos of setting and shutter) divided by the batch size (16). It means that the number of iterations of validation steps: 200 / 16 = 12. Validation\_steps is total number of steps (batches of samples) to validate before stopping. When the model is trained it should be saved with save\_weights. Now, when the model is dissembled it can be run. It can be seen that after 19 epochs the validation accuracy is 0.7083, it shows the ability of the model to generalize to new data. After running the code and saving the model it’s time to check its accuracy on the new testing photos. It is possible through scoring code. After running this code with the new 700 photos of setting and shutter, got a classification accuracy of 82%.

Results

After running the code and saving the model it’s time to check its accuracy on the new testing photos. It is possible through scoring code. After running this code with the new 700 photos of setting and shutter, we got a classification accuracy of 82%.

The plotted graph shows the dependence of accuracy and validation accuracy on the number of epochs during the testing.

This graph shows that the accuracy is increasing as per epoch the highest accuracy we achieved at 32nd epoch 0.9%. After that the value of accuracy is not such good enough.

Problem faced:-

1. Collecting the large amount of data from various site like flaticon.com, goggle images, iconfinder.com.

So, learnt to make a scrapper to download the icons from the various URLs at one go.

2. After renaming all the files there is a space creates in the name of each file which is not acceptable by the model .So, used a python script to remove the spaces.

3. Accuracy of the model is not improving still the loss remains high in each epoch in order to remove this we create multiple layers used three convolution layer, activation layer, pooling layer this improved the accuracy a lot .

1.2 Multi-Classification Convolutional Neural Network Model

Multi-classification model is used by doing slight changes in previous model of binary classification. The layer structure is same as binary but the only difference is in last layer. It need to change the loss function to categorical\_crossentrophy and the class mode to categorical the last output layer activation function changed to softmax and there should be four layers in the last layer units=4 as As it has four classes to predict. Here we took loss function to categorical\_crossentrophy as we have multiple classes to classify. For drawing rectangles around the detected object here comes the role of Opencv. Used opencv canny method for edge detection according to the edges it draws the rectangles using skimage regionprops method .Crop out these rectangle using PIL imagedraw method and save every crop image into one folder . With the help of python dictionary labelled the classes .The predict function return the numpy array of values as the class having high probability to lie and set the threshold to >0.6 as the highest value within the array and its index will be returned. Predict for every image in the folder. Using UIAutomator testing framework stored the x and y coordinates of the predicted icon into the object and call the module as per required action.

Results

Image with grayscale.

Image showing edges.

Image showing detected rectangles.

Problem Faced:

1. Rectangles on the image are not clear as it contained overlapping rectangles.

Before After

This issue is removed by extracting all the overlapping rectangles lies inside one another remove them and print the outermost rectangle.

2. The model is predicting the value which is having probability less than 0.6 showing wrong result thus to remove these value we convert the module (predict\_classes()) to (predict()) gives the exact probability lies under 0 to 1 on the basis of that we set the threshold to >0.6 and remove the unwanted predict values.

After changes

It gives me the numpy array with the class having higher probability to fall.

Conclusion

In this project, to tackle the problem of not being able to retrieve the screen details in OpenGL applications a machine learning based solution is implemented. The machine learning model being discussed above is Multi-classification convolutional neural network model which is trained on considerable amount of pictures with various icons in them. The extensive training of model helps in determining various icons on the screen which we can feed to UI automator. This all task is done is just a jiffy of time that it makes it all a hassle free task which in turn can be used by the testers to analyse using dump information of various layouts. With further improving of the dataset, the accuracy can be increased but for the time being, the model works well in almost all of the cases.

Reference

https://www.pyimagesearch.com/2018/07/19/opencv-tutorial-a-guide-to-learn-opencv/

https://pillow.readthedocs.io/en/3.1.x/reference/ImageDraw.html

https://scikit-image.org/docs/dev/api/skimage.measure.html#skimage.measure.regionprops

https://medium.com/@ksusorokina/image-classification-with-convolutional-neural-networks-496815db12a8

https://www.geeksforgeeks.org/introduction-to-tensorflow/

https://cv-tricks.com/tensorflow-tutorial/keras/

https://github.com/xiaocong/uiautomator

Future Scope

Binary Classification Convolutional Neural Network Model

Shopping cart icon that we can see on various e-commerce sites will be detect by the model when the device is connected as this model will get train by adequate amount of cart images and the after learning predict for the element.