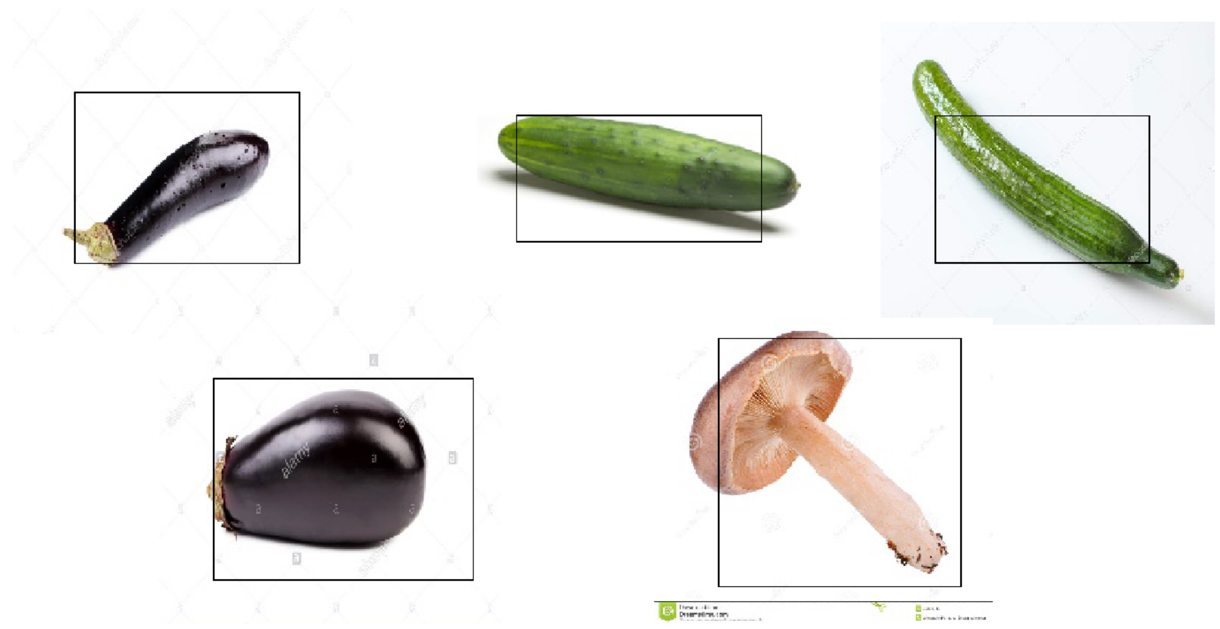
Getting Started With Bounding Box Regression In TensorFlow

Bounding box regression could be your first wonderful step in the world of object detection.

Shubham Panchal
Sep 15 · 4 min read ★



Wanna get started with Object Detection? Any ML learner could like to see nice bounding boxes around an object in an image (at least for me!). We'll now learn a basic concept in Object Detection called Bounding Box Regression. That's easy and can work in cases where you would like to draw a single box on an image. Object detectors like YOLO (You Only Look Once) also use it internally with some changed parameters.

We'll implement a Bounding Box regression model in TensorFlow with Keras API. So, let's start!

Also, all the code required for this project is available on the GitHub repo ~>

https://github.com/shubham0204/Bounding_Box_Regression_TF

Don’t forget to see the Google Colab notebook for this article!

Google Colaboratory

Edit description

colab.research.google.com

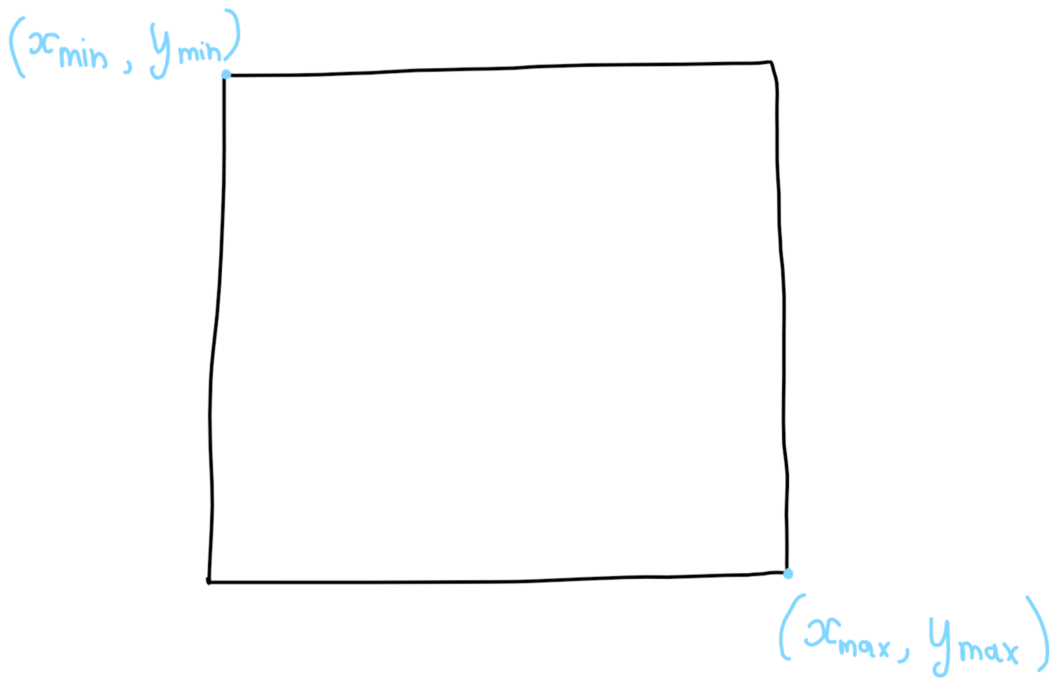
Get me a Dataset! From Kaggle.com?

A Dataset from Kaggle?

We’ll be using the Image Localization Dataset from Kaggle.com by Muhammad Buyukkinaci.

The dataset contains 373 images from three classes (cucumbers, eggplant and mushroom) with their bounding box annotations in XML files. Our goal is to parse the images and normalize them. Also, parse the 4 bounding box coordinates from the XML files.

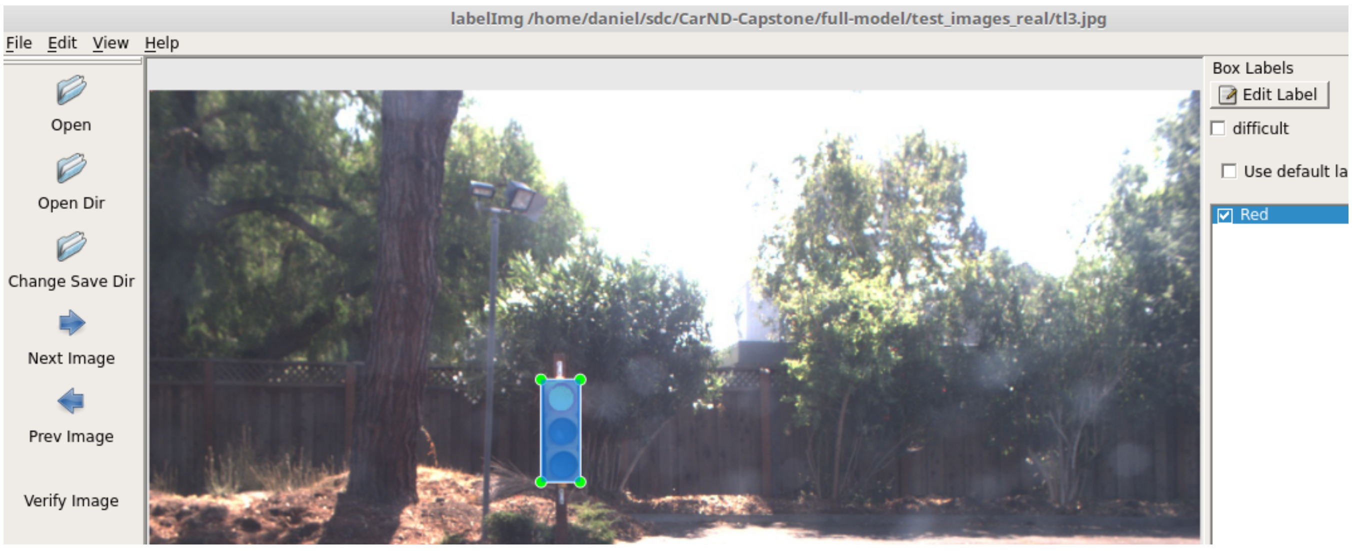
For those who are confused with the bounding box coordinates, here's a diagram for that,



Bounding box coordinates

Want to make your dataset?

Yes, that's possible with LabelImg. It's a great tool for drawing bounding boxes over images and saving them to the PASCAL-VOC format quickly.



Source

Let’s process the data!

First, we’ll process the images. Using the `glob` package, we list all the files which have an extension of `*.jpg` and process them one by one.

```python

input_dim = 228

from PIL import Image , ImageDraw
import os
import glob

images = []
image_paths = glob.glob( 'training_images/*.jpg' )
for imagefile in image_paths:
    image = Image.open( imagefile ).resize( ( input_dim , input_dim ))
    image = np.asarray( image ) / 255.0
    images.append( image )

```

bbox_regress_1.py hosted with ❤ by GitHub view raw

Next, we process the XML annotations. The annotations are in PASCAL-VOC format. We use the `xmltodict` package to transform the XML files to a Python `dict` object.

```python

import xmltodict
import os

bboxes = []
classes_raw = []
annotations_paths = glob.glob( 'training_images/*.xml' )
for xmlfile in annotations_paths:
    x = xmltodict.parse( open( xmlfile , 'rb' ) )
    bndbox = x[ 'annotation' ][ 'object' ][ 'bndbox' ]
    bndbox = np.array([ int(bndbox[ 'xmin' ]) , int(bndbox[ 'ymin' ]) , int(bndbox[ 'xmax' ]) ,
    bndbox2 = [ None ] * 4
    bndbox2[0] = bndbox[0]
    bndbox2[1] = bndbox[1]
    bndbox2[2] = bndbox[2]
    bndbox2[3] = bndbox[3]
    bndbox2 = np.array( bndbox2 ) / input_dim
    bboxes.append( bndbox2 )
    classes_raw.append( x[ 'annotation' ][ 'object' ][ 'name' ] )

```

bbox_regress_2.py hosted with ❤ by GitHub view raw

Now, we assemble all the variables together to prepare the final training and testing datasets.

```

from sklearn.preprocessing import LabelBinarizer
from sklearn.model_selection import train_test_split

boxes = np.array( bboxes )
encoder = LabelBinarizer()
classes_onehot = encoder.fit_transform( classes_raw )

Y = np.concatenate( [ boxes , classes_onehot ] , axis=1 )
X = np.array( images )

x_train, x_test, y_train, y_test = train_test_split( X, Y, test_size=0.1 )
```

bbox_regress_3.py hosted with ❤ by GitHub view raw

Creating the Keras Model.

We'll first define a custom loss function and metric for our model. The loss function has been devised by me and it uses both Mean Squared Error (MSE) and Intersection over Union (IOU). A metric used to measure our model's accuracy will also output an IOU score.

$$L(x, x') = MSE(x, x') + (1 - IOU(x, x'))$$

My Creation!

Where IOU is the ratio of the intersection of the two boxes and the union of the two boxes.

$$IOU(x, x') = \frac{x \cap x'}{x \cup x'}$$

Intersection Over Union

In Python,

```

input_shape = ( input_dim , input_dim , 3 )
dropout_rate = 0.5
alpha = 0.2

def calculate_iou( target_boxes , pred_boxes ):
    xA = K.maximum( target_boxes[ ... , 0], pred_boxes[ ... , 0] )
    yA = K.maximum( target_boxes[ ... , 1], pred_boxes[ ... , 1] )
    xB = K.minimum( target_boxes[ ... , 2], pred_boxes[ ... , 2] )
    yB = K.minimum( target_boxes[ ... , 3], pred_boxes[ ... , 3] )
    interArea = K.maximum( 0.0 , xB - xA ) * K.maximum( 0.0 , yB - yA )
    boxAArea = (target_boxes[ ... , 2] - target_boxes[ ... , 0]) * (target_boxes[ ... , 3] - ta
    boxBArea = (pred_boxes[ ... , 2] - pred_boxes[ ... , 0]) * (pred_boxes[ ... , 3] - pred_box
    iou = interArea / ( boxAArea + boxBArea - interArea )
    return iou

def custom_loss( y_true , y_pred ):
    mse = tf.losses.mean_squared_error( y_true , y_pred )
    iou = calculate_iou( y_true , y_pred )
    return mse + ( 1 - iou )

def iou_metric( y_true , y_pred ):
    return calculate_iou( y_true , y_pred )

```

bbox_regress_4.py hosted with ❤ by GitHub view raw

Next, we create our CNN model. We stack some `Conv2D` layers, flatten their output and pass their output through `Dense` layers.

To avoid overfitting, we use `Dropout` in the `Dense` layers and `LeakyReLU` activation layers.

```
num_classes = 3
pred_vector_length = 4 + num_classes

model_layers = [
        keras.layers.Conv2D(16, kernel_size=(3, 3), strides=1, input_shape=input_shape),
    keras.layers.LeakyReLU( alpha=alpha ) ,
    keras.layers.Conv2D(16, kernel_size=(3, 3), strides=1 ),
    keras.layers.LeakyReLU( alpha=alpha ) ,
    keras.layers.MaxPooling2D( pool_size=( 2 , 2 ) ),

    keras.layers.Conv2D(32, kernel_size=(3, 3), strides=1),
    keras.layers.LeakyReLU( alpha=alpha ) ,
    keras.layers.Conv2D(32, kernel_size=(3, 3), strides=1),
    keras.layers.LeakyReLU( alpha=alpha ) ,
    keras.layers.MaxPooling2D( pool_size=( 2 , 2 ) ),

    keras.layers.Conv2D(64, kernel_size=(3, 3), strides=1),
    keras.layers.LeakyReLU( alpha=alpha ) ,
    keras.layers.Conv2D(64, kernel_size=(3, 3), strides=1),
    keras.layers.LeakyReLU( alpha=alpha ) ,
    keras.layers.MaxPooling2D( pool_size=( 2 , 2 ) ),

    keras.layers.Conv2D(128, kernel_size=(3, 3), strides=1),
    keras.layers.LeakyReLU( alpha=alpha ) ,
    keras.layers.Conv2D(128, kernel_size=(3, 3), strides=1),
    keras.layers.LeakyReLU( alpha=alpha ) ,
    keras.layers.MaxPooling2D( pool_size=( 2 , 2 ) ),

    keras.layers.Conv2D(256, kernel_size=(3, 3), strides=1),
    keras.layers.LeakyReLU( alpha=alpha ) ,
    keras.layers.Conv2D(256, kernel_size=(3, 3), strides=1),
    keras.layers.LeakyReLU( alpha=alpha ) ,
    keras.layers.MaxPooling2D( pool_size=( 2 , 2 ) ),

    keras.layers.Flatten() ,

    keras.layers.Dense( 1240 ) ,
    keras.layers.LeakyReLU( alpha=alpha ) ,
    keras.layers.Dense( 640 ) ,
    keras.layers.LeakyReLU( alpha=alpha ) ,
    keras.layers.Dense( 480 ) ,
    keras.layers.LeakyReLU( alpha=alpha ) ,
    keras.layers.Dense( 120 ) ,
    keras.layers.LeakyReLU( alpha=alpha ) ,
    keras.layers.Dense( 62 ) ,
    keras.layers.LeakyReLU( alpha=alpha ) ,

    keras.layers.Dense( pred_vector_length )
    keras.layers.LeakyReLU( alpha=alpha ) ,
]

model = keras.Sequential( model_layers )
model.compile(
        optimizer=keras.optimizers.Adam( lr=0.0001 ),
        loss=custom_loss,
    metrics=[ iou_metric ]
)
```

bbox_regress_5.py hosted with ❤ by GitHub view raw

Training the model

You can train the model until the loss is converging.

```
model.fit(
    x_train ,
    y_train ,
    validation_data=( x_test , y_test ),
    epochs=100 ,
    batch_size=3
)

model.save( 'model.h5')
```

There's more!

Google Colab notebooks on interesting topics

1. Poem Writer AI - A TensorFlow Project which generates Poetry.
2. Classifying movie reviews with LSTM networks.
3. Artificial Neural Networks. Universal Function Approximators?
4. Colorizing Black & White Faces with Auto-Encoders using TF.
5. Chatbot using Seq2Seq LSTM models.
6. Neural Machine Translation (NMT) - Translating English sentences to French sentences.
7. Neural Machine Translation (NMT) - Translating English sentences to Marathi sentences.
8. Melanoma Classification.
9. Object Detection with Bounding Box Regression.

You may find these notebooks in a number of articles on Medium.com. All of them are authored by Shubham Panchal.

ml_colab_notebooks_res.md hosted with ❤ by GitHub view raw

Resources for Machine Learning with Android

Blogs

- Sarcasm Detection using Word Embeddings in Android
- Hands-on With Multiple Linear Regression on Android
- Bayes Text Classification in Kotlin for Android without TensorFlow
- How I made Skinly for Melanoma Detection in Android
- Text Classification in Android with TensorFlow

Android Apps

- Sarcaso
- Skinly for Melanoma
- Spam Classification Android Demo

AndroTF

You can use AndroTF library to easily integrate your image classification models (TFLite) right in your Android app.

GitHub Projects

All projects are available at https://github.com/shubham0204

Note: All the mentioned blogs/projects are authored by Shubham Panchal

android_ml_res.md hosted with ❤ by GitHub view raw

Drawing Bounding boxes on images

Our model is now trained. We'll predict the classes for some test images and save them to our local disk (that's your choice).

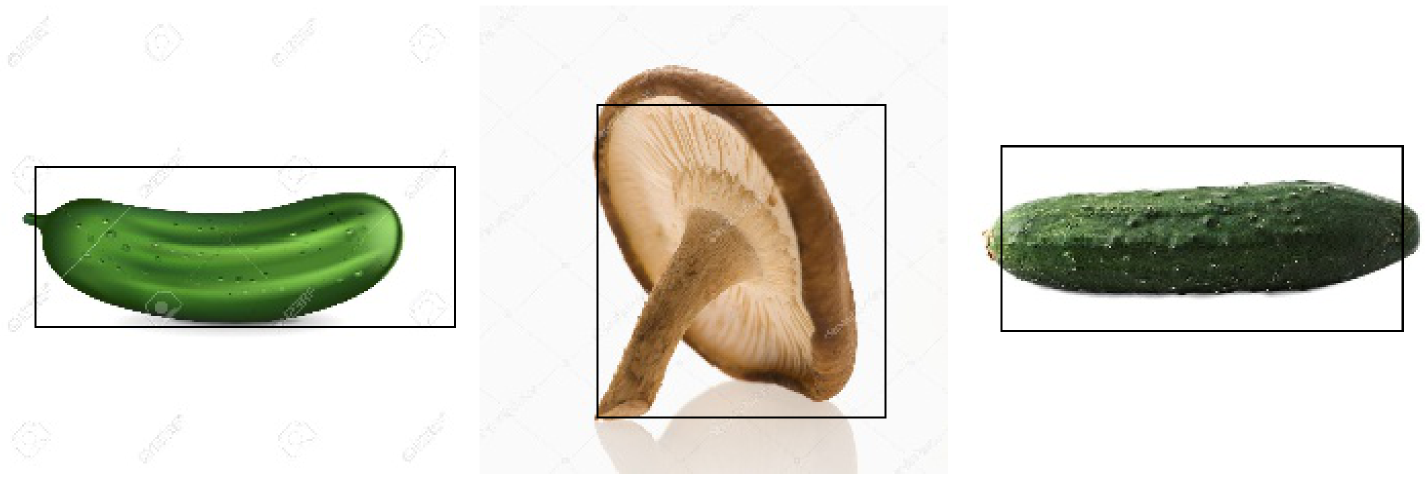
```
!mkdir -v inference_images

boxes = model.predict( x_test )
for i in range( boxes.shape[0] ):
    b = boxes[ i , 0 : 4 ] * input_dim
    img = x_test[i] * 255
    source_img = Image.fromarray( img.astype( np.uint8 ) , 'RGB' )
    draw = ImageDraw.Draw( source_img )
    draw.rectangle( b , outline="black" )
    source_img.save( 'inference_images/image_{}.png'.format( i + 1 ) , 'png' )

```

bbox_regress_6.py hosted with ❤ by GitHub view raw

Here are some images with bounding boxes (I felt wonderful after seeing them!).



Images with Bounding Boxes

My first expressions after seeing the "boxed" images!

To determine the average IOU score over the test dataset and also to calculate the class accuracy, we use the below code.

```python

def calculate_avg_iou( target_boxes , pred_boxes ):
    xA = np.maximum( target_boxes[ ... , 0], pred_boxes[ ... , 0] )
    yA = np.maximum( target_boxes[ ... , 1], pred_boxes[ ... , 1] )
    xB = np.minimum( target_boxes[ ... , 2], pred_boxes[ ... , 2] )
    yB = np.minimum( target_boxes[ ... , 3], pred_boxes[ ... , 3] )
    interArea = np.maximum(0.0, xB - xA ) * np.maximum(0.0, yB - yA )
    boxAArea = (target_boxes[ ... , 2] - target_boxes[ ... , 0]) * (target_boxes[ ... , 3] - ta
    boxBArea = (pred_boxes[ ... , 2] - pred_boxes[ ... , 0]) * (pred_boxes[ ... , 3] - pred_box
    iou = interArea / ( boxAArea + boxBArea - interArea )
    return iou

def class_accuracy( target_classes , pred_classes ):
    target_classes = np.argmax( target_classes , axis=1 )
    pred_classes = np.argmax( pred_classes , axis=1 )
    return ( target_classes == pred_classes ).mean()

target_boxes = y_test * input_dim
pred = model.predict( x_test )
pred_boxes = pred[ ... , 0 : 4 ] * input_dim
pred_classes = pred[ ... , 4 : ]

iou_scores = calculate_avg_iou( target_boxes , pred_boxes )
print( 'Mean IOU score {}'.format( iou_scores.mean() ) )

print( 'Class Accuracy is {} %'.format( class_accuracy( y_test[ ... , 4 : ] , pred_classes ) *
```

bbox_regress_7.py hosted with ❤ by GitHub view raw

That's All

My first impressions regarding bounding box regression were not could but its something you should definitely if you're further going to learn YOLO or SSD object detectors. Thanks and happy Machine Learning.

Machine Learning Object Detection Computer Vision Artificial Intelligence TensorFlow