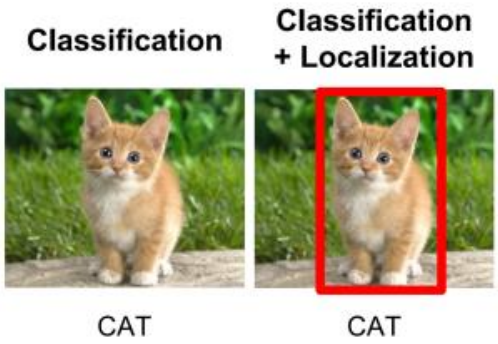
Object Localization and Object Detection

Topics Covered

1. Object Localization Overview
2. Data Labelling for Object Localization and Detection
3. IOU (Intersection Over Union)
4. Mean Average Precision (mAP)
5. Image Augmentation
6. Bounding Box Augmentation
7. Object Localization Using Transfer Learning
8. Summary

Object Localization Overview

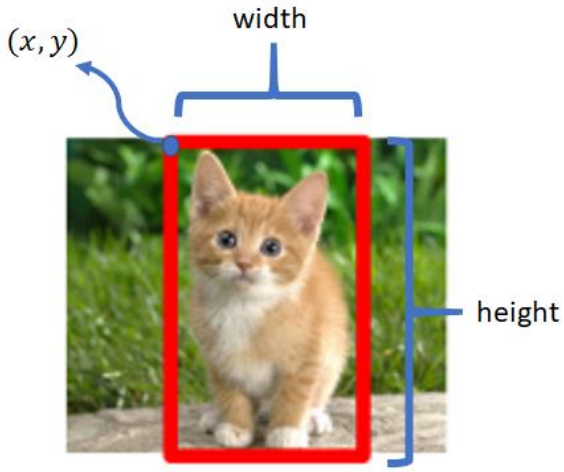
Object localization is the name of the task of “classification with localization”. Namely, given an image, classify the object that appears in it, and find its location in the image, usually by using a bounding-box. In Object Localization, only a single object can appear in the image. If more than one object can appear, the task is called “Object Detection”.



Classification vs Object Localization

Object Localization using DNNs

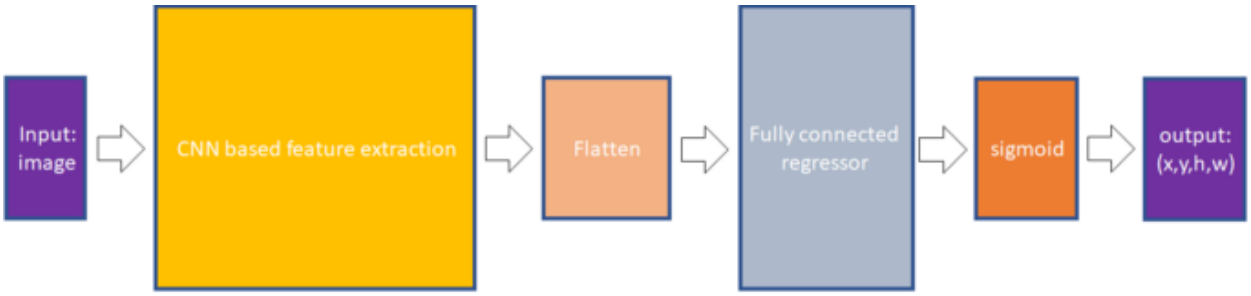
Object Localization can be treated as a regression problem - predicting a continuous value, such as a weight or a salary. For instance, we can represent our output (a bounding-box) as a tuple of size 4, as follows:
(x,y, height, width)
- (x,y): the coordination of the left-top corner of the bounding box
- height: the height of the bounding box
- width: the width of the bounding box



(x,y,height,width) as a bounding-box representation

Hence, we can use a CNN architecture that will output 4 numbers, but what will be our architecture? First, we will notice that these 4 numbers have some constraints: (x.y) must be inside the image and so do x+width and y+height. We will scale the image width and height to be 1.0. To make sure that the CNN outputs will be in the range [0,1], we will use the *sigmoid* activation layer- it will enforce that (x,y) will be inside the image, but not necessarily x+width and y+height. This property will be learned by the network during the training process.

suggested architecture:



Given an image, the network will output a 4 numbers representation bounding box

What about the loss function? the output of a *sigmoid* can be treated as probabilistic values, and therefore we can use binary_crossentropy loss.

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}\log(h_\theta(x^{(i)})) + (1-y^{(i)})\log(1-h_\theta(x^{(i)}))]$$

binary_crossentropy formula

Usually, this loss is being used with binary values as the ground-truth ({0,1}), but it doesn't have to be this way- we can use values from [0,1]. For our use, the ground-truth values are indeed in the range [0,1], since it represents location inside an image and dimensions.

Data Labelling for object localization and detection

There are various labeling approaches. depending on the problem statement, the time frame of the project, and the number of people who are associated with the work.

While internal labeling and crowdsourcing are very common, the terminology can also extend to include novel forms of labeling and annotation that make use of AI and active learning for the task.

The most common approaches for annotation of data are listed below.

In-house data labeling

In-house data labeling secures the highest quality labeling possible and is generally done by data scientists and data engineers hired at the organization.

High-quality labeling is crucial for industries like insurance or healthcare, and it often requires consultations with experts in corresponding fields for proper labeling of data.

As is expected for in-house labeling, with the increase in quality of the annotations, the time taken to annotate increases drastically, resulting in the entire data labeling process and cleaning being very slow.

Crowdsourcing

Crowdsourcing refers to the process of obtaining annotated data with the help of a large number of freelancers registered at a crowdsourcing platform.

The datasets annotated consist mostly of trivial data like images of animals, plants, and the natural environment and they do not require additional expertise. Therefore, the task of annotating a simple dataset is often crowdsourced to platforms that have tens of thousands of registered data annotators.

Outsourcing

Outsourcing is a middle ground between crowdsourcing and in-house data labeling where the task of data annotation is outsourced to an organization or an individual.

One of the advantages of outsourcing to individuals is that they can be assessed on the particular topic before the work has been handed over.

This approach of building up annotation datasets is perfect for projects that do not have much funding, yet require a significant quality of data annotation.

Machine-based annotation

One of the most novel forms of annotation is machine-based annotation. Machine-based annotation refers to the use of annotation tools and automation which can drastically increase the speed of data annotating without sacrificing the quality.

The good news is that recent automation developments in traditional machine annotation tools—using unsupervised and semi-supervised machine learning algorithms—helped significantly reduce the workload on the human labelers.

Unsupervised algorithms like clustering and recently developed semi-supervised algorithms for AI data labeling—like active learning are tools that can reduce annotation times by bounds.

Common types of data labeling

From what we have seen till now, data labeling is all about the task we want a machine-learning algorithm to perform with our data.

For example—

If we want a machine learning algorithm for the task of defect inspection, we feed it data such as images of rust or cracks. The corresponding annotation would be polygons for localization of those cracks or corrosion, and tags for naming them.

Preparation of data with bounding boxes using tool labelimg

For in house labelling we can use LabelImg is a graphical image annotation tool. It is written in Python and uses Qt for its graphical interface.

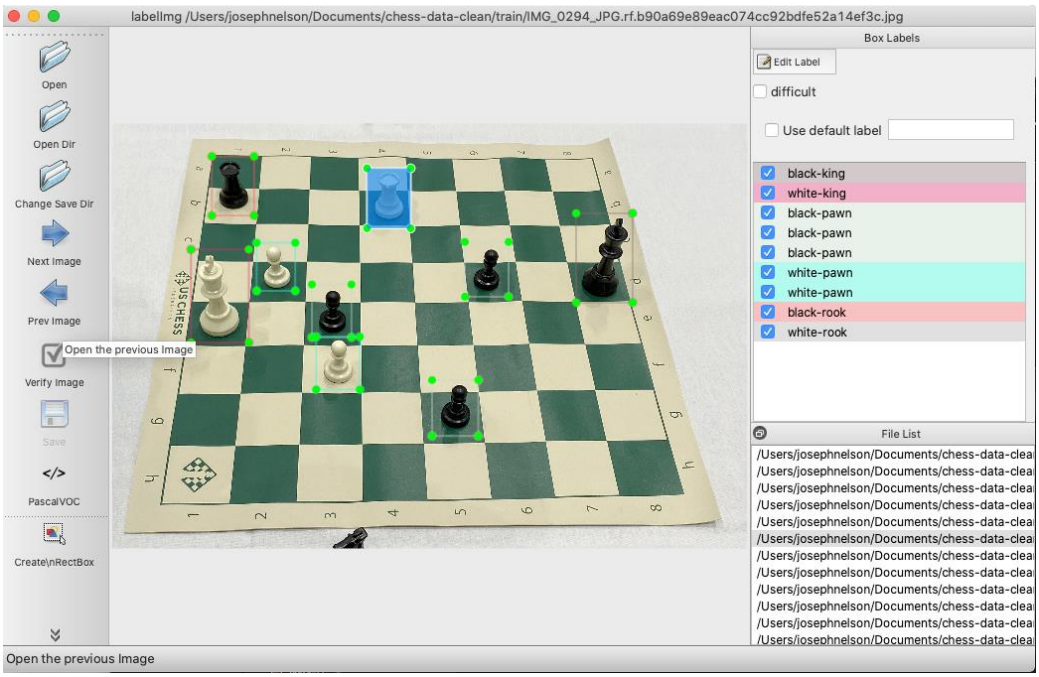
Annotations are saved as XML files in PASCAL VOC format, the format used by ImageNet. Besides, it also supports YOLO and CreateML formats.

Labeling Images with LabelImg

LabelImg supports labelling in VOC XML or YOLO text file format. Thanks to ImageNet, VOC XML is a more universal standard as it relates to object detection whereas various YOLO implementations have slightly different text file formats. Moreover, you can always easily convert from VOC XML to any other format using Roboflow, like VOC XML to COCO JSON.

Open your desired set of images by selecting “Open Dir” on the left-hand side of LabelImg

To initiate a label, type w, and draw the intended label. Then, type ctrl (or command) S to save the label. Type d to go to the next image (and a to go back an image).



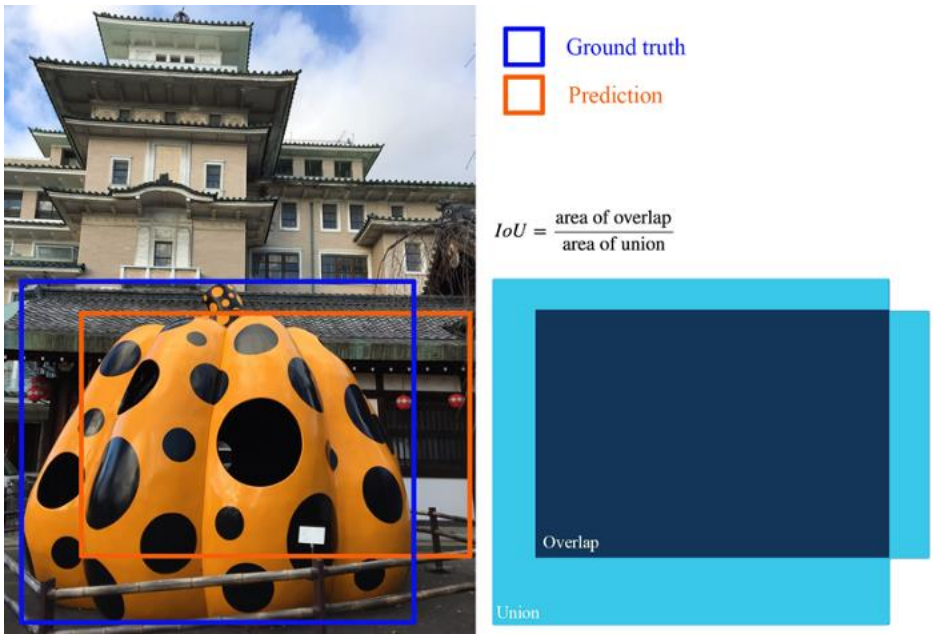
Labeling Best Practices

When labeling images, there are a few tips to bear in mind. Note that these tips are generally true, but can alter based on the context of your problem.

1. Label around the entirety of an object. It is best to include a little bit of non-object buffer than it is to exclude a portion of the object with a rectangular label. So, aim to have boxes that tightly mirror the objects you want to label, but do not cut off part of the objects. Your model will understand edges far better this way.
2. For occluded objects, label them entirely. If an object is out of view due to another object being in front of it, label the object out of view as if you could see its entirety. Your model will begin to understand the true bounds of objects this way.
3. For objects partially out of frame, generally label them. This tip especially depends on your problem, but in general, even a partial object is still an object to be labeled.
4. If outsourcing a labeling job, provide crystal clear instructions. If you're ready to scale up your labeling operations and bring on outside help, be incredibly explicit in your instructions (like mentioning the importance of labeling around an object rather than cutting a portion of the object out of bounds)

IoU (Intersection over union)

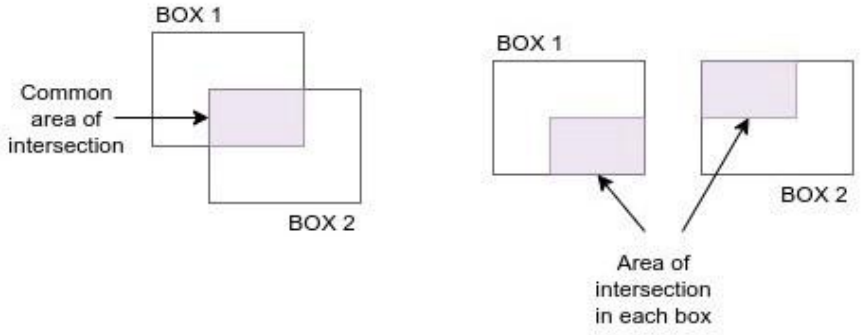
IoU measures the overlap between 2 boundaries. We use that to measure how much our predicted boundary overlaps with the ground truth (the real object boundary). In some datasets, we predefine an IoU threshold (say 0.5) in classifying whether the prediction is a true positive or a false positive.



IoU definition

IOU is mainly used in applications related to object detection, where we train a model to output a box that fits perfectly around an object. For example in the image below, we have a green box, and a blue box. The green box represents the correct box, and the blue box represents the prediction from our model. The aim of this model would be to keep improving its prediction, until the blue box and the green box perfectly overlap, i.e the IOU between the two boxes becomes equal to 1.

If we look at the total area covered by the two boxes, we see that the portion of the intersection is covered in both the boxes, i.e the area of the intersection is included in both **area_box1** and **area_box2.**



Depicting the intersection area being covered in both boxes.

- Since we want to account for the common area of intersection only once, we can subtract the area of intersection we calculated, from the total area of the two boxes.

$$area\_union = area\_box1 + area\_box2 - area\_intersection$$

Area of the union of the boxes

Calculating the IOU

$$IOU = \frac{area\_inter}{area\_union}$$

Formula to calculate IOU

What is the range of values that the IOU can have?

- The IOU of two boxes can have any values between 0 and 1.
- In case there are 2 boxes that do not intersect, the area of their intersection would be 0, and therefore the IOU would also be 0.
- In case there are 2 boxes that completely overlap, the area of the intersection would be equal to the area of their union, and therefore the IOU would be 1.

Coding a function for IOU in python:

- For the same image displayed above, lets write a function in python to calculate the IOU of the green and the blue box.

```
def IOU(box1, box2):
    x1, y1, x2, y2 = box1
    x3, y3, x4, y4 = box2
    x_inter1 = max(x1, x3)
    y_inter1 = max(y1, y3)
    x_inter2 = min(x2, x4)
    y_inter2 = min(y2, y4)
    width_inter = abs(x_inter2 - x_inter1)
    height_inter = abs(y_inter2 - y_inter1)
    area_inter = width_inter * height_inter
    width_box1 = abs(x2 - x1)
    height_box1 = abs(y2 - y1)
    width_box2 = abs(x4 - x3)
    height_box2 = abs(y4 - y3)
    area_box1 = width_box1 * height_box1
    area_box2 = width_box2 * height_box2
    area_union = area_box1 + area_box2 - area_inter
    iou = area_inter / area_union
    return iou
```

Function in python to implement IOU

- The function IOU takes in 2 boxes, box1 and box2 as input. The data in each box is a list containing[x1, y1, x2, y2], which is the top left, and bottom right coordinates.
- We find the area of the intersection, followed by the area of the union, as described earlier.
- The abs function is an inbuilt function in python to calculate the modulus. This ensures we never end up with a negative width or height.
- The IOU returned is a value of type float, that lies between 0 and 1.

Before moving to mAP lets refer precision and recall

Precision measures how accurate is your predictions. i.e. the percentage of your predictions are correct.

Recall measures how good you find all the positives. For example, we can find 80% of the possible positive cases in our top K predictions.

Here are their mathematical definitions:

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

TP = True positive

TN = True negative

FP = False positive

FN = False negative

For example, in the testing for cancer:

$$Precision = \frac{TP}{\text{total positive results}}$$

$$Recall = \frac{TP}{\text{total cancer cases}}$$

MEAN AVERAGE PRECISION (mAP)

- The evaluation metric used in object recognition tasks is 'mAP'. It is a number from 0 to 100 and higher values are typically better.
- Each bounding box will have a score associated (likelihood of the box containing an object).
- Based on the predictions, a precision-recall curve (PR curve) is computed for each class by varying the score threshold.
- The average precision (AP) is the area under the PR curve. First the AP is computed for each class, and then averaged over the different classes. The end result is the mAP.

the mAP of the object detection model is calculated according to the below equation.

$$mAP = \frac{1}{n}\sum_{k=1}^{k=n} AP_k$$

$$AP_k = \text{the AP of class } k$$

$$n = \text{the number of classes}$$

Variations Among mAP

In most of the research papers, these metrics will have extensions like mAP iou = 0.5, mAP iou = 0.75, mAP small, medium, large. In this, we will clearly demonstrate what it actually means.

- **mAP iou=0.5** represents the model has used 0.5 threshold value to remove unnecessary bounding boxes, it is the standard threshold value for most of the models.
- **mAP iou=0.75** represents the model has used 0.75 threshold value, By using this we can get accurate results by removing bounding boxes with less than 25% of the intersection with ground truth image.
- **mAP small** represents the model has given mAP score based on smaller objects in the data.
- **mAP large** represents the model has given mAP score based on larger objects in the data.

Image Augmentation

Image augmentation is a technique of altering the existing data to create some more data for the model training process. In other words, it is the process of artificially expanding the available dataset for training a deep learning model.

In this picture, the image on the left is only the original image, and the rest of the images are generated from the original training image.

Since all these images are generated from training data itself we don't have to collect them manually. This increases the training sample without going out and collecting this data. Note that, the label for all the images will be the same and that is of the original image which is used to generate them.

Different Image Augmentation Techniques

Now let us look at different image augmentation techniques

Image rotation

one of the most commonly used augmentation techniques is image rotation. Even if you rotate the image, the information on the image remains the same. A cat is a cat, even if you see it from a different angle.

Hence, we can use this technique to increase the size of our training data by creating multiple images rotated at different angles.

Image Shifting

The next image augmentation technique is image shifting. By shifting the images, we can change the position of the objects in the image and hence give more variety to the model. Which eventually can result in a more generalized model.

Image shift is a geometric transformation that maps the position of every object in the image to the new location of the final output image. So if an object is at position x,y in the original image, it gets shifted to new position X, Y in the new image as shown below. Where dx and dy are the respective shifts along with the different directions.

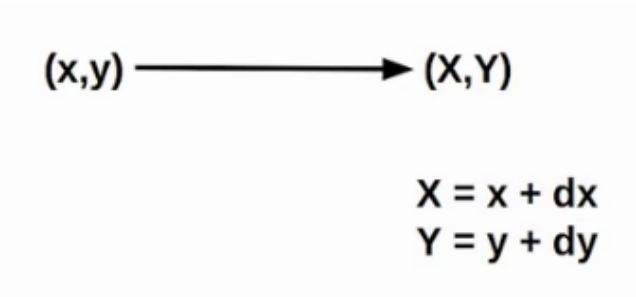


Image Flipping

The next technique is Image flipping. Flipping can be considered as an extension of rotation. it allows us to flip the image in the Left-Right direction as well as the Up-Down direction.

The leftmost image here is the original image of the training set and the remaining two images are the flipped images.

Image Noising

Another popularly used image augmentation technique is, Image Noising where we add noise to the image. This technique allows our model to learn how to separate the signal from the noise in the image. This also makes our model more robust to changes in the image

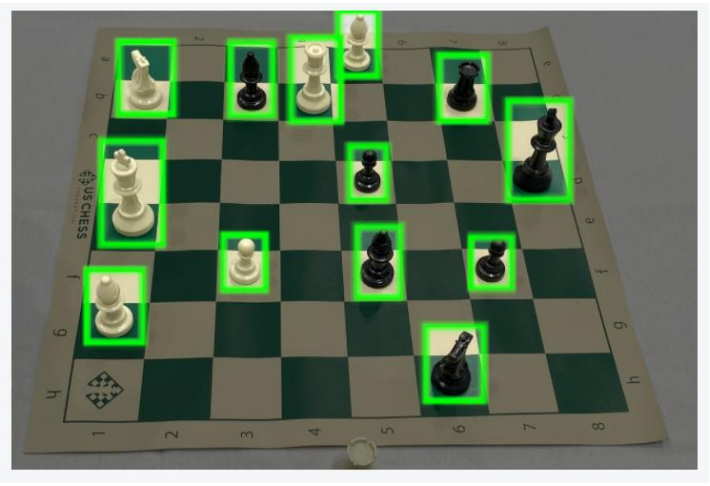
Image Blurring

Let's understand one more augmentation technique, Image blurring. Images come from different sources and hence the quality of images will not be the same from each source. Some images might be of very high quality and others must be very bad. In such scenarios we can blur the original images, this will make our model more robust to the quality of the image being used in the test data.

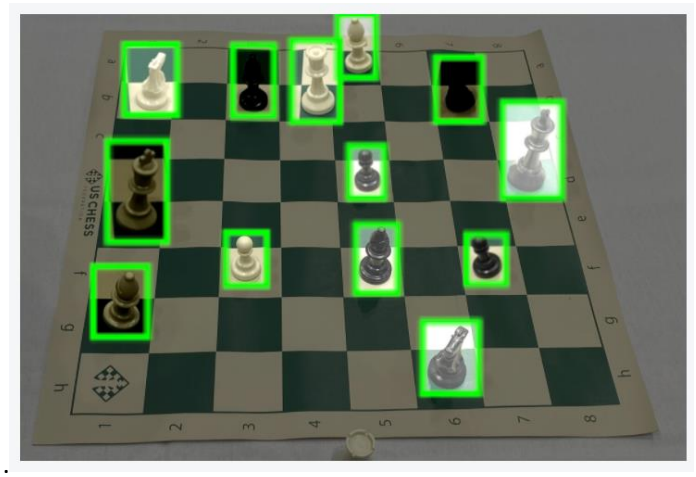
So these are some commonly used image augmentation techniques. This article was a short introduction to the image augmentation techniques used for feature engineering. If you are looking for more details and the implementation.

Bounding Box Augmentation

Bounding box augmentation takes this idea and goes one step further: what if we altered the content of an image, but only the content within a given bounding box? For example, vary the brightness or darkness of an object relative to its background. Or, perhaps blur an object relative to its background for tasks that are often capturing rapidly moving objects.



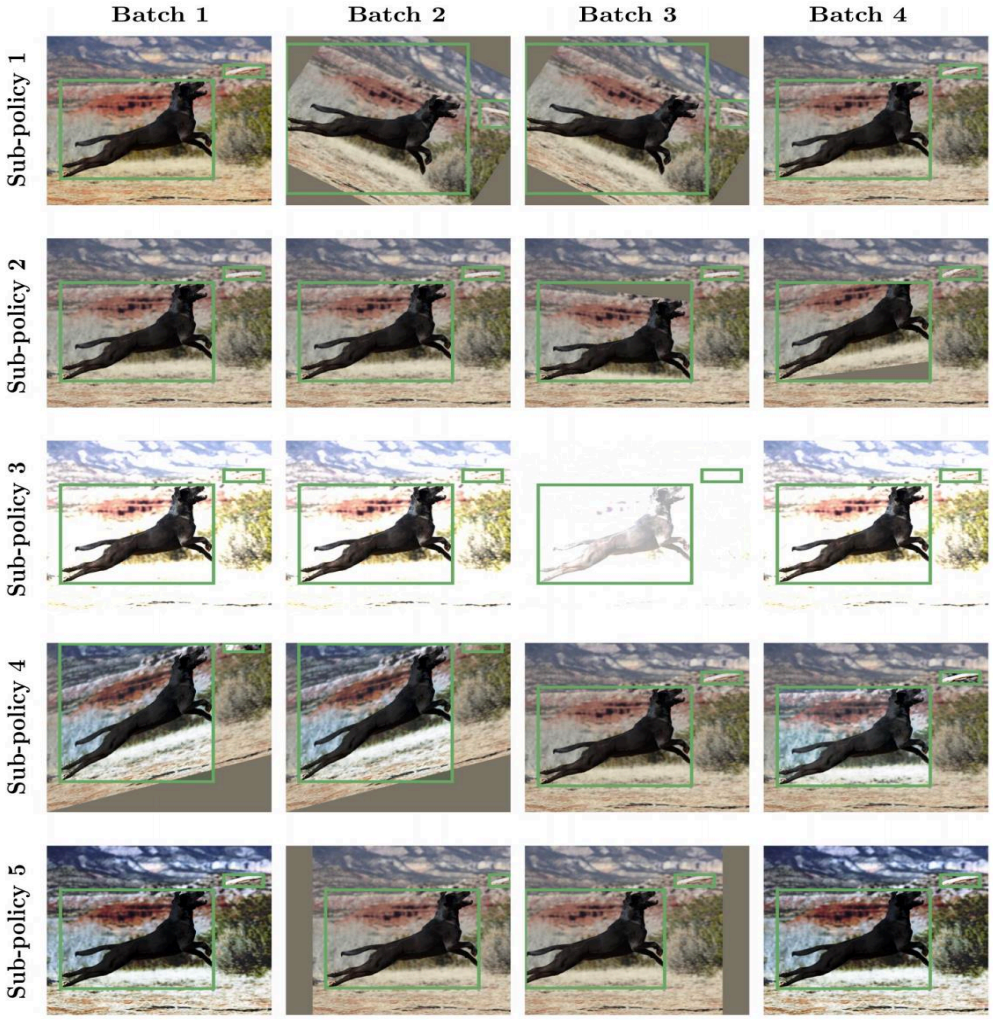
An original image



Bounding box only augmentation that has randomly horizontally flipped and altered the brightness of objects. Note that each individual bounding box receives its own settings.

The Genesis of Bounding Box Level Augmentation

Google researchers introduces the idea of using bounding box only augmentation to create optimal data for their models. In this paper, researchers showed bounding box only modifications generated systemic improvements, especially for models that were small datasets.



Sub-policy 1. (Color, 0.2, 8), (Rotate, 0.8, 10)
Sub-policy 2. (BBox_Only_ShearY, 0.8, 5)
Sub-policy 3. (SolarizeAdd, 0.6, 8), (Brightness, 0.8, 10)
Sub-policy 4. (ShearY, 0.6, 10), (BBox_Only_Equalize, 0.6, 8)
Sub-policy 5. (Equalize, 0.6, 10), (TranslateX, 0.2, 2)

Summary

- Object localization is the name of the task of “classification with localization”. Namely, given an image, classify the object that appears in it, and find its location in the image, usually by using a bounding-box
- For in house labelling, we can use LabelImg is a graphical image annotation tool. It is written in Python and uses Qt for its graphical interface.
- Image augmentation is a technique of altering the existing data to create some more data for the model training process.
- In bounding box augmentation, we alter the content of an image, but only the content within a given bounding box.