

# Object Detection with the BigML Dashboard

The BigML Team

Version 1.0



MACHINE LEARNING MADE BEAUTIFULLY SIMPLE

**Copyright© 2024, BigML, Inc., All rights reserved.**

[info@bigml.com](mailto:info@bigml.com)

BigML and the BigML logo are trademarks or registered trademarks of BigML, Inc. in the United States of America, the European Union, and other countries.

BigML Products are protected by US Patent No. 11,586,953 B2; 11,328,220 B2; 9,576,246 B2; 9,558,036 B1; 9,501,540 B2; 9,269,054 B1; 9,098,326 B1, NZ Patent No. 625855, and other patent-pending applications.

This work by BigML, Inc. is licensed under a [Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License](https://creativecommons.org/licenses/by-nc-nd/4.0/). Based on work at <http://bigml.com>.

*Last updated June 7, 2024*

# About this Document

This document provides a comprehensive description of how to perform object detection using the BigML [Dashboard](#). Learn how to use the BigML Dashboard to configure, visualize, and interpret this [supervised](#) model and use it to make predictions.

This document assumes that you are familiar with:

- Sources with the BigML Dashboard. The BigML Team. April 2022. [\[5\]](#)
- Datasets with the BigML Dashboard. The BigML Team. April 2022. [\[4\]](#)

To learn how to use the BigML Dashboard to build other [supervised](#) predictive models read:

- Classification and Regression with the BigML Dashboard. [\[2\]](#)

To learn how to use the BigML Dashboard to build other unsupervised models read:

- Cluster Analysis with the BigML Dashboard. The BigML Team. January 2022. [\[3\]](#)
- Association Discovery with the BigML Dashboard. The BigML Team. June 2016. [\[1\]](#)
- Topic Modeling with the BigML Dashboard. The BigML Team. November 2016. [\[6\]](#)

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Understanding Object Detection</b>	<b>3</b>
2.1	Image Data for Object Detection	3
2.2	Training for Object Detection	4
2.3	Object Detection Evaluation Metrics	9
2.3.1	IOU - Intersection Over Union	9
2.3.2	Prediction Score	9
2.3.3	True Positives, False Positives and False Negatives	9
2.3.4	Precision, Recall and PR Curve	10
2.3.5	F1--Score	10
2.3.6	Average Precision and Mean Average Precision	10
<b>3</b>	<b>Annotating Images</b>	<b>12</b>
3.1	Regions	12
3.1.1	Pixel Coordinates and Normalized Coordinates	12
3.1.2	JSON Notations of Regions	14
3.2	Annotating Images with JSON Files	15
3.3	Annotation Interface	16
3.4	Annotating Images Interactively	21
3.5	Cropping Images By Regions	25
<b>4</b>	<b>Creating Object Detection Deepnets</b>	<b>28</b>
4.1	Creating Object Detection Deepnets with 1-Click	28
4.2	Object Detection Deepnet Configuration Options	29
4.2.1	Objective Fields	30
4.2.2	Default Regions Value	31
4.2.3	Training Duration	31
4.2.4	Maximum Iterations	31
<b>5</b>	<b>Visualizing Object Detection Deepnets</b>	<b>33</b>
5.1	Introduction	33
5.2	Visualization Page Layout	34
5.2.1	Performance Panel	35
5.2.2	Image Results	36
5.2.2.1	Image Closeup View	38
5.2.3	Class List	39
5.3	Inspecting Predictions	41
<b>6</b>	<b>Object Detection Evaluations</b>	<b>46</b>
6.1	Introduction	46
6.2	Creating Object Detection Evaluations	48
6.3	Visualizing Object Detection Evaluations	49



6.3.1	Main Control . . . . .	50
6.3.2	Evaluation View - Class Detail . . . . .	50
6.3.2.1	Selection . . . . .	51
6.3.2.2	Chart . . . . .	51
6.3.2.3	Metrics . . . . .	53
6.3.2.4	Image Preview . . . . .	53
6.3.3	Evaluation View - Summary . . . . .	54
6.3.3.1	Selection . . . . .	54
6.3.3.2	Stats . . . . .	55
6.3.3.3	Table . . . . .	55
<b>7</b>	<b>Object Detection Predictions</b>	<b>56</b>
7.1	Introduction . . . . .	56
7.2	Creating Object Detection Predictions . . . . .	58
7.2.1	Single Predictions . . . . .	58
7.2.2	Batch Predictions . . . . .	63
	<b>List of Figures</b>	<b>71</b>
	<b>Glossary</b>	<b>73</b>
	<b>References</b>	<b>74</b>

# Introduction

Object detection is a computer vision technique for locating objects of interest in images. When human beings look at images, they can recognize and locate objects of interests almost instantly. The goal of object detection is to simulate such human vision, or to simulate this human intelligence using computers.

Object detection starts with image data. With the support of images and annotation by composite sources, BigML object detection workflows start with BigML SOURCE, which is the same for all BigML workflows. Object detection workflows also require other BigML resources, including DATASET, DEEPNET, EVALUATION, PREDICTION and BATCH PREDICTION.

Image annotation in object detection involves class labels and bounding boxes. Annotations are implemented in terms of a BigML field type, *regions*. The value of a regions field specifies a list of bounding boxes and their class labels in an image.

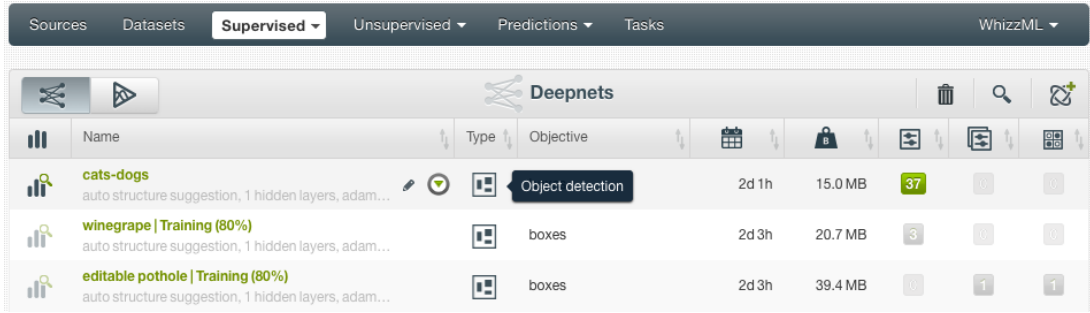
Object detection is a supervised learning method. At BigML, an object detection model is a deepnet trained with a regions field as its objective field.

Figure 1.1 shows the icon used to represent an object detection resource in BigML.



Figure 1.1: Object detection icon

To access object detection models, go to the deepnet list view. Click on the DEEPNET section under the SUPERVISED tab of the main menu of the BigML Dashboard. This shows all deepnets created so far. Object detection deepnets are denoted by a distinctive icon in the **Type** column. The deepnet list view also shows the dataset used to create each object detection deepnet, its **Name** and the parameters configured to create the deepnet, its **Objective** field, **Age** (time elapsed since its creation), **Size**, and the number of **Evaluations**, **Predictions** and **Batch Predictions** that have been created using the object detection deepnet. The SEARCH menu option in the top right menu of the deepnet list view can be used to search a deepnet by name and by configuration options.



Name	Type	Objective						
<b>cats-dogs</b> auto structure suggestion, 1 hidden layers, adam...	Object detection	boxes	2d 1h	15.0 MB	37	0	0	0
<b>winegrape   Training (80%)</b> auto structure suggestion, 1 hidden layers, adam...	boxes	boxes	2d 3h	20.7 MB	3	0	0	0
<b>editable pothole   Training (80%)</b> auto structure suggestion, 1 hidden layers, adam...	boxes	boxes	2d 3h	39.4 MB	0	1	1	1

Figure 1.2: Object detection deepnets in deepnet list view

This document provides a comprehensive description of BigML object detection, including how training data is uploaded and annotated, how object detection models can be created by 1-click, how to configure different parameters, and object detection visualization. An object detection model can be evaluated for its performance. If performance goals are satisfied, the object detection model can be used to predict, in both single predictions and batch predictions.

# Understanding Object Detection

Object detection is a supervised learning method. It can be used to predict or detect objects in given images, showing the classes and locations of those objects. To train an object detection model, the image data for training has to be annotated with object information. The object information is represented by a BigML field type **regions**. After the image data is annotated, it's used to train a BigML DEEPNET as the object detection model. Deepnet is the BigML resource for deep neural networks.

The following sections cover image data annotation, deepnet training and evaluation metrics.

## 2.1 Image Data for Object Detection

Object detection is performed on image data. To train an object detection model, the image data needs to have information about the objects in the images. This way, the models can learn how to detect those objects in new images.

The act of providing information about images is called *annotation*. Annotation is indispensable in object detection. The process of annotation is to mark what objects are where in an image. It is done by drawing a bounding box for every object in the image and assigning a class label to the bounding box.

A *bounding box* is a rectangle drawn in an image, outlining the object of interest within the image. Because a bounding box surrounds an object, it acts as a reference to the location of the object. A *class label*, sometimes just *label*, is the category or type of the object inside the bounding box. There may be many classes of objects in the image. A class label denotes an instance of a class. Please keep in mind that annotation is the information about an image, it won't become a part of an image. While a bounding box is drawn in an image, it's imaginary, and it's not actually added to the image.

BigML provides a field type, *regions* to support object detection annotation. The value of a regions field specifies bounding boxes and their class labels in an image. In essence, a regions field uses a 5-tuple (a list of 5 elements) to represent one bounding box and its class label:

```
[label, x, y, width, height]
```

where `label` is the class label of the object, `x` and `y` are the coordinates of the center of the bounding box, `width` and `height` the width and the height of the bounding box, respectively.

BigML supports interactive image annotation on the Dashboard. Users can upload images to create image composite sources, then create regions fields and add bounding boxes and class labels by using the annotation interface in composite sources.

BigML also supports image annotation by using JSON data. When a collection of images are uploaded, it may come with a JSON file specifying objects in the images, referenced by image file names.

## 2.2 Training for Object Detection

Object detection is a significant feature on top of image classification. While image classification determines whether an image belongs to a certain class, object detection goes further – not only it shows where an object is in the image, but also it can show where instances of objects from multiple classes are located in the image.

Deepnet is the BigML resource for deep neural networks, which are excellent at machine learning with images. When creating a deepnet from a dataset whose objective field is a regions field, it will train a convolutional neural network (CNN) as the object detection model. CNN is a special type of deep neural networks.

This is similar to image classification and regression. For simplicity, we only use image classification for comparison here.

When creating a deepnet from a dataset which has a categorical field as its objective field, BigML also trains a CNN as the image classification model. But they have different deepnet outputs.

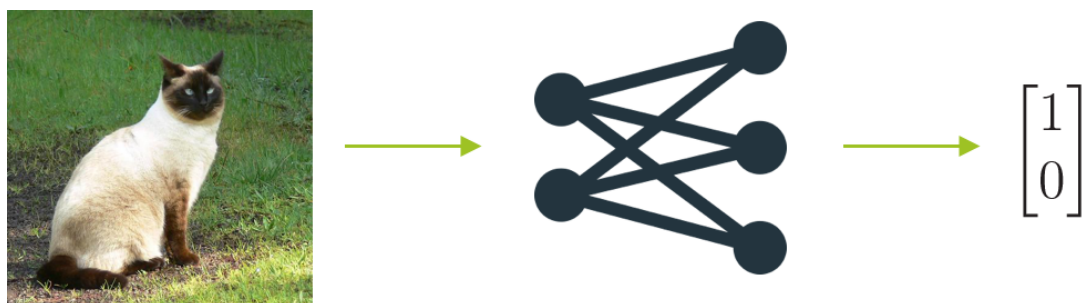


Figure 2.1: Training a classification deepnet

We use an image dataset as an example, which has two classes, *cat* and *dog*. Its classification deepnet output takes the form of  $\begin{bmatrix} p_1 \\ p_2 \end{bmatrix}$  where  $p_1$  is the probability of the image belonging to the *cat* class,  $p_2$  the probability of belonging to the *dog* class. The number of classes determines the vector dimension. In [Figure 2.1](#), the image is labeled *cat*, so the output is  $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$ . For an image labeled *dog*, its output is  $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$  ([Figure 2.2](#)).

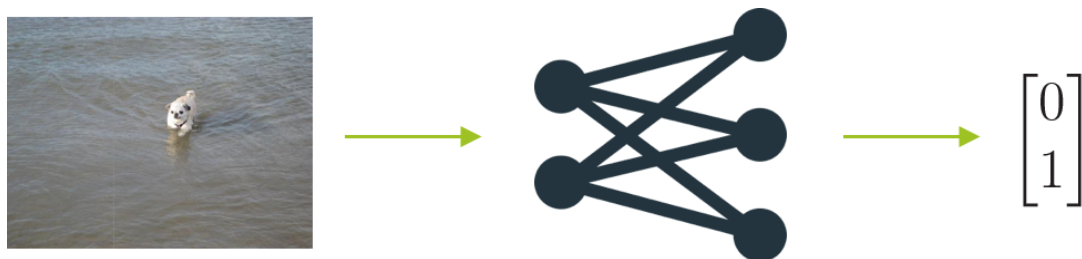


Figure 2.2: Also training a classification deepnet

Keep in mind that the outputs shown here are for training. They are derived from image labels, so the probabilities are either 1 or 0. When it comes to prediction, an image classification output may look like  $\begin{bmatrix} 0.89 \\ 0.11 \end{bmatrix}$ , which means the image is classified as *cat* with a probability of 0.89.

Suppose we use the same images for object detection. First, we need to annotate them with bounding boxes and class labels. Their deepnet outputs become more complex, 7-element vectors instead of 2-element ones. For the image in [Figure 2.1](#), after it's annotated, its object detection deepnet output would be

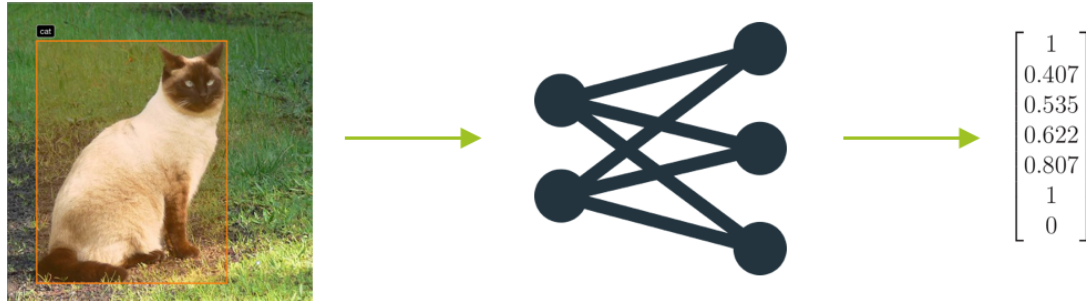


Figure 2.3: Training an object detection deepnet

Now the deepnet output is in the form of

$$\begin{bmatrix} p_c \\ r_x \\ r_y \\ r_w \\ r_h \\ p_1 \\ p_2 \end{bmatrix}$$

where  $p_c$  is the probability that the image has at least one object,  $(r_x, r_y)$  are the coordinates of the center of the bounding box,  $(r_w, r_h)$  are the width and height of the bounding box, and  $p_1$  is the probability of the object belonging to the class of *cat*,  $p_2$  the probability of belonging to the class of *dog*. Basically, the output describes a bounding box and its class label.  $p_c$  is significant and useful, because when it's 0 that means there is no object in the image and all other elements in the vector don't matter, hence saving computation and time. The dimension of the vector will increase along with the number of classes.

Specifically in [Figure 2.3](#),  $p_c$  is 1 indicating there is at least one object in the image.  $(0.407, 0.535)$  is the center of the bounding box annotated on the object – a cat.  $(0.622, 0.807)$  are the width and height of the bounding box.  $p_1$  is 1 and  $p_2$  is 0 because it's a cat.

When the picture in [Figure 2.2](#) is used for object detection, after it's annotated with the bounding box and class label, its output would be what's shown in [Figure 2.4](#).  $p_c$  is 1 indicating objects in the image, the four numerical values  $(0.568, 0.416, 0.112, 0.203)$  specify the location of the annotated bounding box, and  $p_2$  is 1 because the object is a dog.

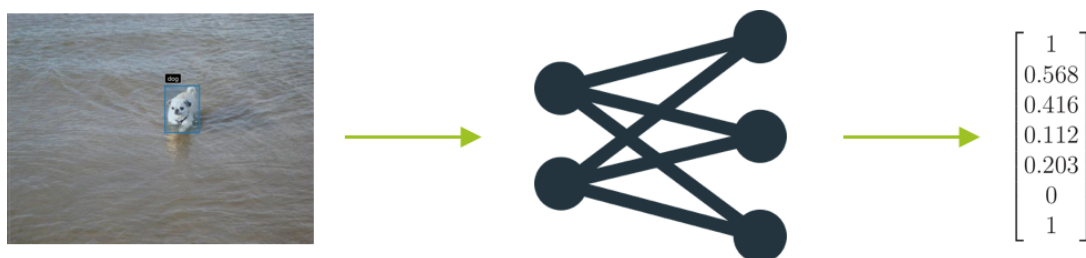


Figure 2.4: Also training an object detection deepnet

Again, those outputs are derived from image annotations and are used for deepnet training.

Now we understand the different outputs used for the training of classification and object detection deepnets. An Object detection model's objective field contains class labels as well as bounding boxes, which are represented by numerical coordinates. So object detection is essentially a supervised learning that combines classification and regression.

Apparently what have been described so far are only good for single-object images. In object detection deepnet training, every image is divided into grid cells.

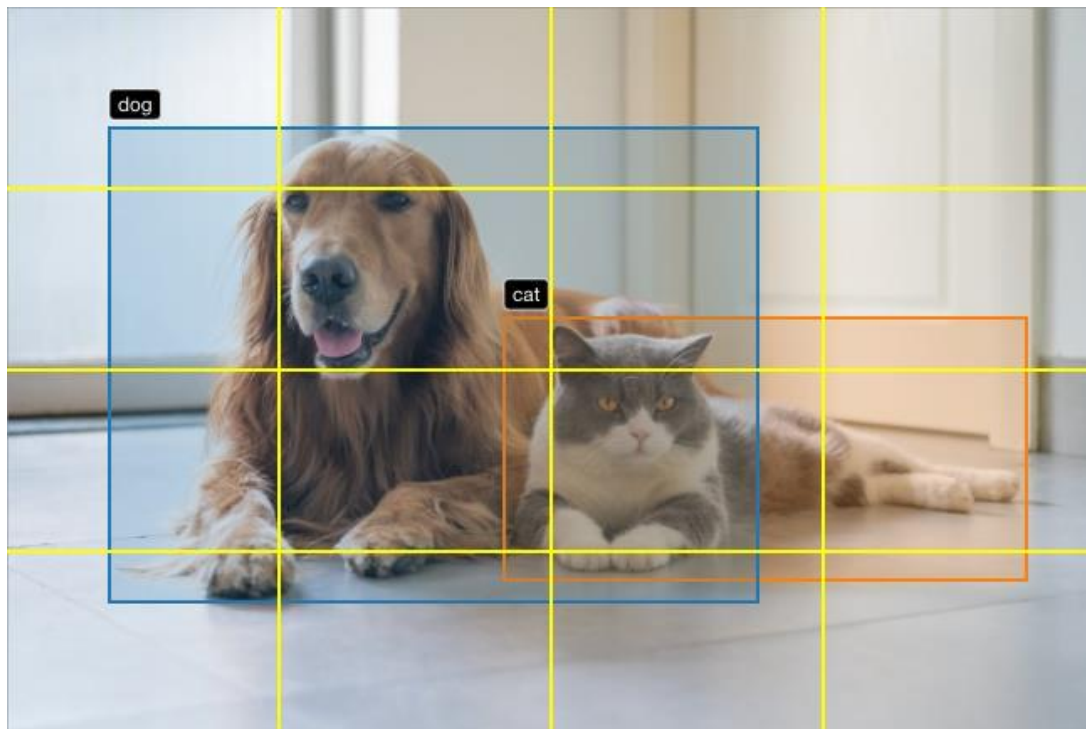


Figure 2.5: A training image divided by grids

Here in Figure 2.5, 4x4 grids are used for illustration. In real training, higher grids such as 13x13 or 19x19 could be used.

Each grid cell contributes to the deepnet output. For the upper right grid cell in Figure 2.5, there is no object in it, so its output is

$$\begin{bmatrix} 0 \\ * \\ * \\ * \\ * \\ * \\ * \\ * \end{bmatrix}$$

where 0 means no object in the cell, and \* indicates that those values don't matter. In fact, all cells except two in the image have the above vector as outputs. Only two cells are considered containing objects.

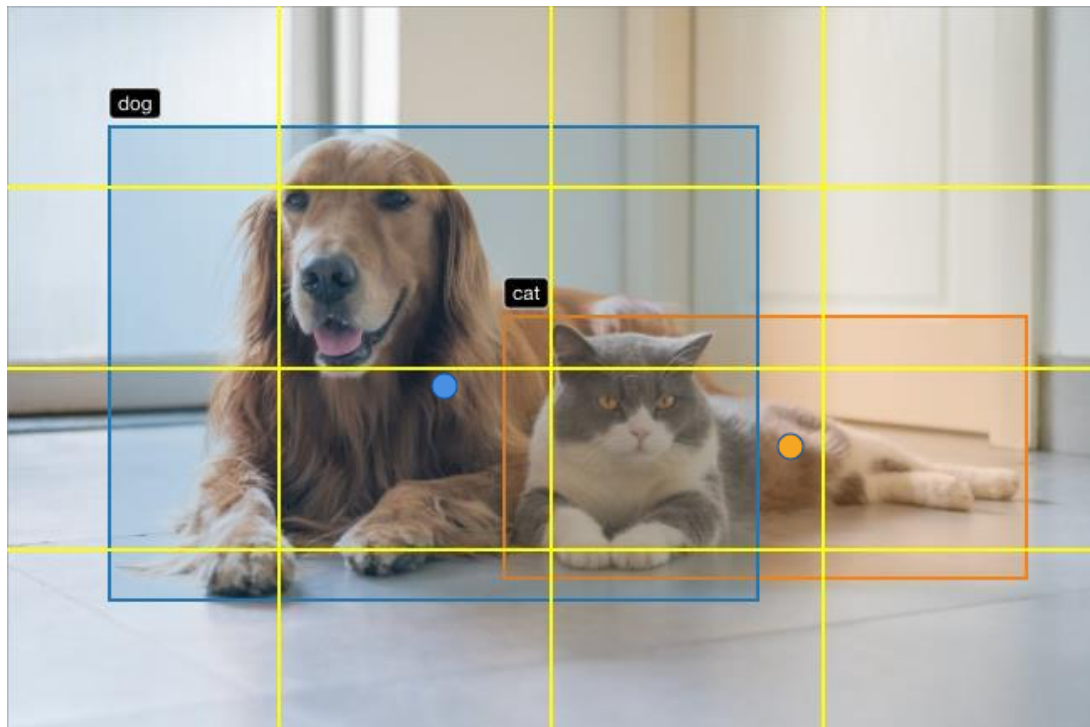


Figure 2.6: A training image divided by grids

Even though the bounding box of an object can span many grid cells, it is considered to belong to the grid cell that contains the center of the bounding box. In Figure 2.6, the blue dot is the center of the bounding box of the dog, so the dog bounding box belongs to the grid cell that's on the 3rd row and the 2nd column of the grids.

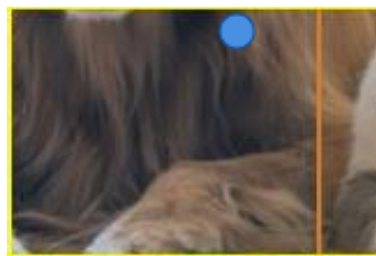


Figure 2.7: A grid cell with the center of a bounding box

We are essentially looking at the grid cell shown in Figure 2.7. Its output is

$$\begin{bmatrix} 1 \\ 0.611 \\ 0.093 \\ 2.387 \\ 2.620 \\ 0 \\ 1 \end{bmatrix}$$

where (0.611, 0.093) is the center of the dog bounding box. The coordinates are normalized, so they are relative to the width and height of the grid cell - the top left vertex of the cell is (0, 0) and the bottom right (1, 1). (2.387, 2.720) are the width and height of the bounding box of the dog. They are bigger than 1



because they occupy more than one grid – they are respectively relative to the width and height of the grid cell.  $p_2$  is 1 because it's a dog.



Figure 2.8: Another grid cell with the center of a bounding box

For the cat bounding box, its center is in the grid cell that's on the 3rd row and the 3rd column, shown in Figure 2.8. Its output is

$$\begin{bmatrix} 1 \\ 0.912 \\ 0.426 \\ 1.926 \\ 1.435 \\ 1 \\ 0 \end{bmatrix}$$

where (0.912, 0.426) are the coordinates of the cat bounding box center. (1.926, 1.435) are the width and height of the bounding box.  $p_1$  is 1 because the bounding box is labeled as *cat*.

The rest of the grid cells don't contain bounding boxes, so their outputs are all

$$\begin{bmatrix} 0 \\ * \\ * \\ * \\ * \\ * \\ * \end{bmatrix}$$

Combining all of them together, the deepnet output for this image is a 4x4x7 matrix. 4x4 are the grid size, and 7 is the vector dimension.

Sometimes, more than one object may share a grid cell, that is, the centers of their bounding boxes fall into the same cell. This situation becomes less likely with the increase of the grid size. When it does happen, each object in the grid cell would have a vector as its output as if it were the only object in the cell. Then the deepnet output of the grid cell is a vector created by concatenating the vectors from all objects in the cell. Say, if there is a grid cell shared by one dog and one cat in an image of the aforementioned dataset, its output will be a vector of 14 elements.

That's how an object detection deepnet constructs its output from the images and their annotations. Training of the deepnet is the process of learning all its neural network weights from the images (as the inputs) and the corresponding outputs.

For more information, references and further reading about object detection, please visit [its wikipedia entry](https://en.wikipedia.org/wiki/Object_detection)<sup>1</sup>.

<sup>1</sup>[https://en.wikipedia.org/wiki/Object\\_detection](https://en.wikipedia.org/wiki/Object_detection)

## 2.3 Object Detection Evaluation Metrics

Evaluation is essential in object detection because we want to know the performance of the object detection deepnets. Then we can compare them and select the best performing one.

Because object detection uses bounding boxes to locate objects, there should be an evaluation metric that measures how good a predicted bounding box is.

### 2.3.1 IOU - Intersection Over Union

Intersection over union, or IOU, is a metric that evaluates the accuracy of a predicted bounding box. It does so by measuring the degree of overlap between the predicted bounding box and the ground truth bounding box. A ground truth bounding box is the bounding box from the annotation. Unlike a predicted bounding box, a ground truth one doesn't come from the model and is usually manually drawn and is considered to surround the object accurately.

As its name suggests, IOU is defined as the value of dividing the area of intersection by the area of union between the ground truth and predicted bounding boxes.

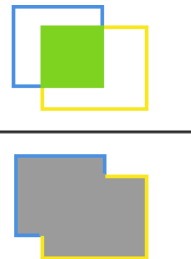
$$IOU = \frac{\text{area of intersection}}{\text{area of union}} = \frac{\text{green area}}{\text{gray area}}$$


Figure 2.9: IOU - Intersection over Union

More illustratively, it's the ratio of the green area over the gray area in Figure 2.9, for instance, supposing the yellow bounding box is the ground truth while the blue box is the predicted. An IOU value is between 0 and 1, inclusively. When IOU is 1, there is a perfect match between the predicted and the ground truth bounding boxes. When IOU is 0, there is no overlap between the bounding boxes which means the predicted box missed the ground truth completely. Generally, we consider a prediction with an IOU over 0.5 a good prediction, while an IOU greater than 0.9 denotes an excellent one.

Once an IOU threshold is set, only predicted bounding box which IOU is greater than the threshold is considered correct, otherwise incorrect. In other words, an IOU threshold decides whether a predicted bounding box is correct or not.

### 2.3.2 Prediction Score

In addition to IOU, the probability of the model classifying an object into the correct class also plays a role in determining whether the prediction is correct or not. Even when a predicted bounding box achieves a high IOU, it is still incorrect if it identifies the object as a wrong class.

Therefore, in object detection, a prediction score, or simply score, is the product of the probability of the predicted bounding box containing an object and the probability of the object belonging to the predicted class.

Prediction score depends on the IOU threshold. After an IOU threshold is set, a prediction score threshold can be set on top of the IOU threshold in order to decide whether a prediction is correct.

### 2.3.3 True Positives, False Positives and False Negatives

Once a prediction score threshold is set, we can measure the deepnet model performance by counting correct and incorrect predictions.

In classification problems, a confusion matrix is employed to count the correct predictions as well as the errors by the model. Object detection also employ these terms, but their definitions are slightly different.

- True Positive (TP): a prediction having the correct class and its prediction score equal to or greater than the threshold
- False Positive (FP): a prediction having either an incorrect class or its prediction score lower than the threshold
- False Negative (FN): an object having a ground truth bounding box but not predicted by the model

Note that, unlike in classification, there is no True Negative (TN). In object detection, the vast majority of areas in an image could be nothing, so the model could predict numerous “nothing” in an image. That’s all true negatives, so the TN count could dominate in metrics, which would make the metrics not useful.

### 2.3.4 Precision, Recall and PR Curve

Like in classification, precision in object detection measures the degree of correctness in a model's predictions, and recall measures the ability of the model detecting all objects. Both metrics are calculated with respect to a certain class.

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{TP}{\text{All Predictions}}$$

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{TP}{\text{All Objects}}$$

Also like in classification, there is a trade-off between precision and recall. In other words, precision and recall are typically inversely related – increasing precision may reduce recall and increasing recall may reduce precision. By lowering the prediction score threshold, we can increase the number of true positives, therefore improving recall. But precision suffers. And vice versa: increasing the score threshold normally reduces the number of false positive, therefore improving precision, but recall gets worse.

Given a class, the Precision-Recall Curve (PR Curve) is a plot of precision and recall at varying values of prediction score. The area under the PR Curve, also known as PR AUC, is a good measure for both high Precision and high Recall.

### 2.3.5 F1-Score

The F1-Score, also called F-measure, is the balanced harmonic mean between Precision and Recall (Subsection 2.3.4). Its value ranges between 0 and 1, inclusively.

$$\text{F1-Score} = \frac{2}{\text{Precision}^{-1} + \text{Recall}^{-1}} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Due to the inverse relationship between precision and recall, an F1-Score is often a useful measure for a model's overall performance because a poor number in either precision or recall would result in a low F1-Score.

Since both Precision and Recall depend on the prediction score threshold, the F1-Score does too. The Maximum F1-Score is the greatest F1-Score among all score threshold values.

### 2.3.6 Average Precision and Mean Average Precision

Average precision (AP) is defined as the area under the Precision-Recall Curve.

In practice, AP is often calculated by averaging precision values at several levels of recall. For instance, an 11-point AP is the average of precision values at 11 equally spaced recall levels, which are (0.0, 0.1, 0.2, ..., 0.9, 1.0).

Precision is calculated with regard to a given class. So for every class in the model, there is an AP. To better measure a model's overall performance, mean average precision (mAP) is defined as the average of APs of all classes in a model.

Please note that AP is calculated at a given IOU threshold. So when the IOU threshold changes, all APs change, so does mAP.

## Annotating Images

As stated in the previous chapters, object detection starts with BigML composite sources. Images are uploaded to create composite sources. They also can be annotated as composite sources.

In object detection specifically, the process of annotation is to mark what objects are where in an image. It is done by drawing a bounding box for every object in the image and assigning a class label to the bounding box.

### 3.1 Regions

Object detection annotation is supported by a field type, *regions*. It represents a bounding box that surrounds an object in an image. It consists of the class label and the coordinates of a bounding box.

A regions value can be specified with a tuple list, or 5-tuple such as

```
[label, xmin, ymin, xmax, ymax]
```

where `label` is the class to which the object belongs, `(xmin, ymin)` are the coordinates of the top left vertex of the bounding box, `(xmax, ymax)` the coordinates of the bottom right vertex.

#### 3.1.1 Pixel Coordinates and Normalized Coordinates

A digital image is made up of rows and columns of pixels. The coordinates of a pixel are the pair of integers giving the column number and the row number of the pixel, usually denoted as  $(x, y)$ , where  $x$  is the column number and  $y$  the row number. The columns are numbered from left to right, and rows numbered from top to bottom, both starting with 0. These coordinates are also called pixel-based coordinates, or simply pixel coordinates.

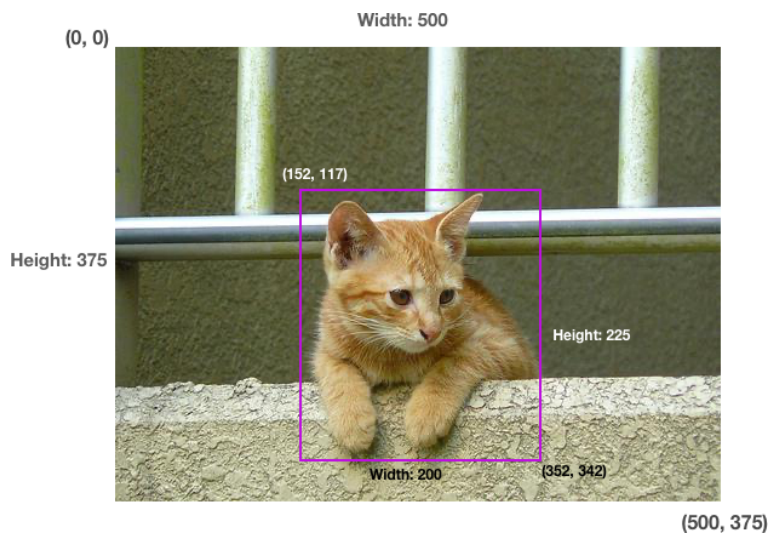


Figure 3.1: Pixel coordinates

In [Figure 3.1](#), the top left vertex of the image is (0, 0), the starting point of the coordinates. Its bottom right vertex is (500, 375), with its width 500 and height 375. The bounding box surrounding the cat is from (152, 117) to (352, 342), with its width 200 and height 225.

Normalized coordinates are generated by dividing the pixel coordinates by the pixel width and pixel height of the image, respectively. Given the pixel coordinates as  $(x, y)$ ,  $w$  the image width in pixel,  $h$  the image height in pixel, then their normalized coordinates would be  $(\frac{x}{w}, \frac{y}{h})$ .

Normalized coordinates are relative to the image dimensions. The top left vertex remains (0, 0), but the bottom right vertex is always (1, 1). One feature of normalized coordinates is that since they are ratios (relative to dimensions), even when the image size is changed (enlarged or shrunk), the coordinates stay the same. Normalized coordinates are numbers between 0 and 1, inclusively. [Figure 3.2](#) shows the normalized pixels of the same image and bounding box in [Figure 3.1](#).

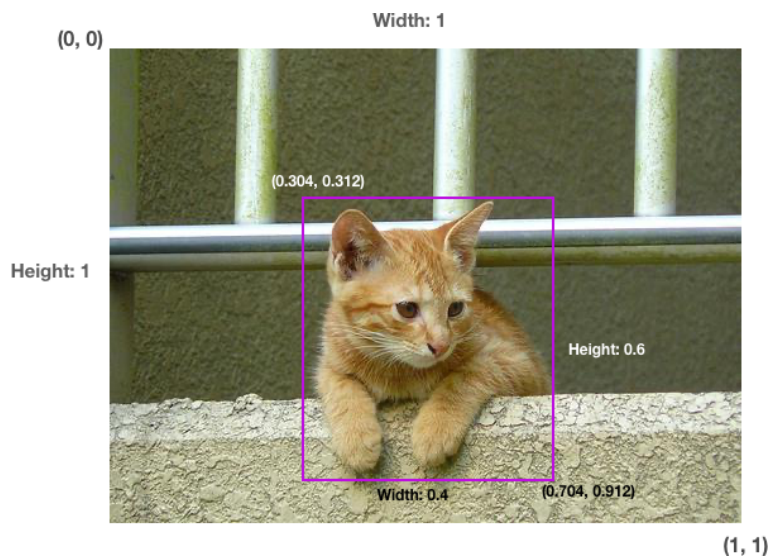


Figure 3.2: Normalized coordinates

Using pixel coordinates, the image can be annotated with a regions value of

```
["cat", 152, 117, 352, 342]
```

where (152, 117) are the pixel coordinates of the top left vertex of the bounding box, and (352, 342) the coordinates of the bottom right vertex. Using normalized coordinates, the regions value becomes

```
["cat", 0.304, 0.312, 0.704, 0.912]
```

where (0.304, 0.312) is the top left vertex of the bounding box, (0.704, 0.912) the bottom right vertex.

BigML accepts both pixel and normalized coordinates in object detection annotations. For internal representation, BigML uses normalized coordinates.

**Note:** If given a coordinate that is greater than 1 but is not an integer, BigML will round the coordinate to the nearest integer and use the integer as the pixel coordinate.

### 3.1.2 JSON Notations of Regions

It is also possible to input regions in composite sources using values in a variety of JSON formats. Each region must be a list of 5-tuples. In each 5-tuple, its first element is the region's class label, and the rest are the bounding box coordinates.

For instance,

```
[["cat", 20, 150, 250, 300]]
```

is one region, while

```
[["cat", 20, 150, 250, 300], ["dog", 225, 50, 500, 280]]
```

are two regions in one image. Adding the regions field name and the file path of the image to reference it, the notation is a JSON object, also known as a JSON map,

```
{"file": "a.png", "box": [["cat", 20, 150, 250, 300], ["dog", 225, 50, 500, 280]]}
```

Again, BigML also accepts normalized coordinates, so the JSON map above could also be

```
{"file": "a.png", "box": [["cat", 0.04, 0.375, 0.5, 0.75], ["dog", 0.45, 0.125, 1, 0.7]]}
```

The coordinates in the regions field values can be specified in different formats using a JSON map with any of these key collections:

1. (label, xmin, ymin, xmax, ymax) where (xmin, ymin) is the top left vertex of the region, while (xmax, ymax) the bottom right vertex. This is the canonical tuple representation. If no keys are used in a JSON map, the coordinates are assumed to be in this format.
2. (label, xmin, ymin, width, height) where (xmin, ymin) is the top left vertex of region, while (width, height) the width and height of the region, respectively.
3. (label, xmax, ymax, width, height) where (xmax, ymax) is the bottom right vertex of region, while (width, height) the width and height of the region, respectively.
4. (label, x, y, width, height) where (x, y) is the center of the region, while (width, height) the width and height of the region, respectively.
5. (label, xcenter, ycenter, width, height) which is a synonym of 4, with xcenter for x and ycenter for y.

For example, the following are two representations of the same region (the backslash \ is the line continuation mark, meaning the two lines should be read as a single line):

```
{"file": "a.png", "box": [["cat", 20, 150, 250, 300]]}
```

```
{"file": "a.png", "box": [{"label": "cat", "xmin": 20, "ymin": 150, \
                          "xmax": 250, "ymax": 300}]}
```

The same region can also be specified as:

```
{"file": "a.png", "box": [{"label": "cat", "x": 135, "y": 225, \
                          "width": 230, "height": 150}]}
```

## 3.2 Annotating Images with JSON Files

BigML supports image annotation by using JSON files.

An image archive (zip or tar) can include a JSON file specifying the annotations of objects in the images, referenced by image file names. The content of the JSON file is an array of JSON objects (aka JSON maps). Each JSON map represents an image, and contains an image file name and its regions field. The value of the regions field is a list of either 5-element JSON arrays or JSON maps. The JSON arrays or maps represent the regions in the image. In previous sections, a 5-element JSON array was referred to as a 5-tuples, or a 5-element list. Their formats for regions are specified in [Subsection 3.1.2](#).

For example, this is the content of a JSON file specifying the annotation of one image:

```
[
  {
    "file": "3364_00086.jpg",
    "boxes": [["dog", 167, 39, 421, 271]]
  }
]
```

The following is equivalent to above:

```
[
  {
    "file": "3364_00086.jpg",
    "boxes": [
      {
        "label": "dog",
        "xmin": 167,
        "ymin": 39,
        "xmax": 421,
        "ymax": 271
      }
    ]
  }
]
```

This is the content of a JSON file specifying the annotations of two images:

```
[
  {
    "file": "3364_00086.jpg",
    "boxes": [
      {
        "label": "dog",
        "xmin": 167,
        "ymin": 39,
        "xmax": 421,
        "ymax": 271
      }
    ]
  },
  {
    "file": "3364_00092.jpg",
    "boxes": [
      {
        "label": "dog",
        "xmin": 23,
        "ymin": 144,
        "xmax": 261,
        "ymax": 292
      }
    ]
  }
]
```



```

    },
    {
      "label": "cat",
      "xmin": 225,
      "ymin": 51,
      "xmax": 500,
      "ymax": 283
    }
  ]
}
]

```

After an image archive containing a JSON file is uploaded, it will create a `table+image` composite source, which is not editable. For more information, please refer to section Table+Image Composite Sources of the [Sources with the BigML Dashboard](#)<sup>1</sup>[5].

A `table+image` composite source can be converted to an editable image composite source using the **CONVERT TO EDITABLE COMPOSITE** menu action, as shown in [Figure 3.3](#). Then you can view all annotations and make changes to them if necessary.

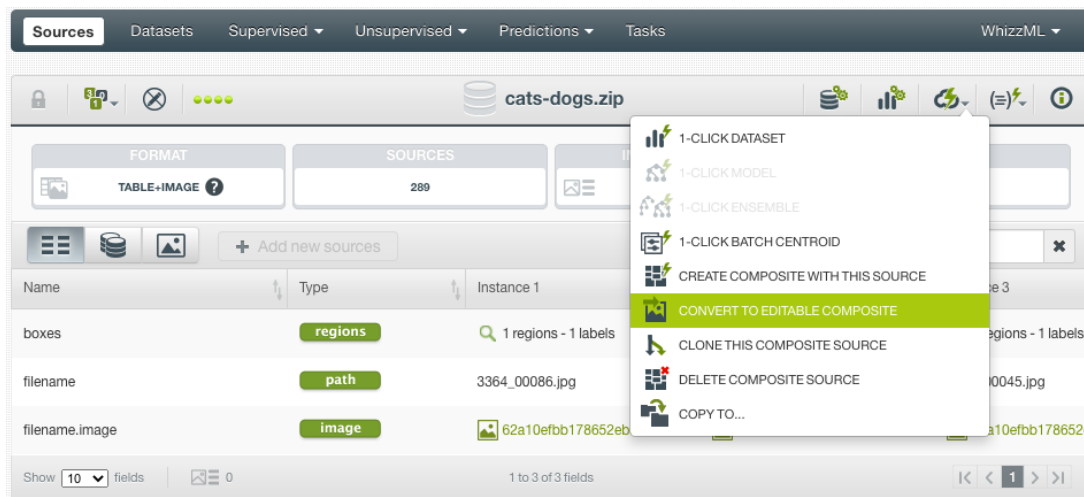


Figure 3.3: Converting a `table+image` composite to an editable composite

### 3.3 Annotation Interface

BigML provides an annotation interface for image composite sources, so that users can annotate images interactively.

The interface is accessed by going to the **Images** view of an open image composite source. It has to be open so it is modifiable. Closed composite sources are not modifiable.

If the image composite source doesn't have a `regions` field, one can be added. As shown in [Figure 3.4](#), under the **Images** view there is a **Label field** drop-down list. Click on the plus sign **+** to add a new label field. Give the new field a name, and select the field type *Regions* from the drop-down list. Then click **Add** to add the new regions field.

<sup>1</sup>[https://static.bigml.com/pdf/BigML\\_Sources.pdf](https://static.bigml.com/pdf/BigML_Sources.pdf)

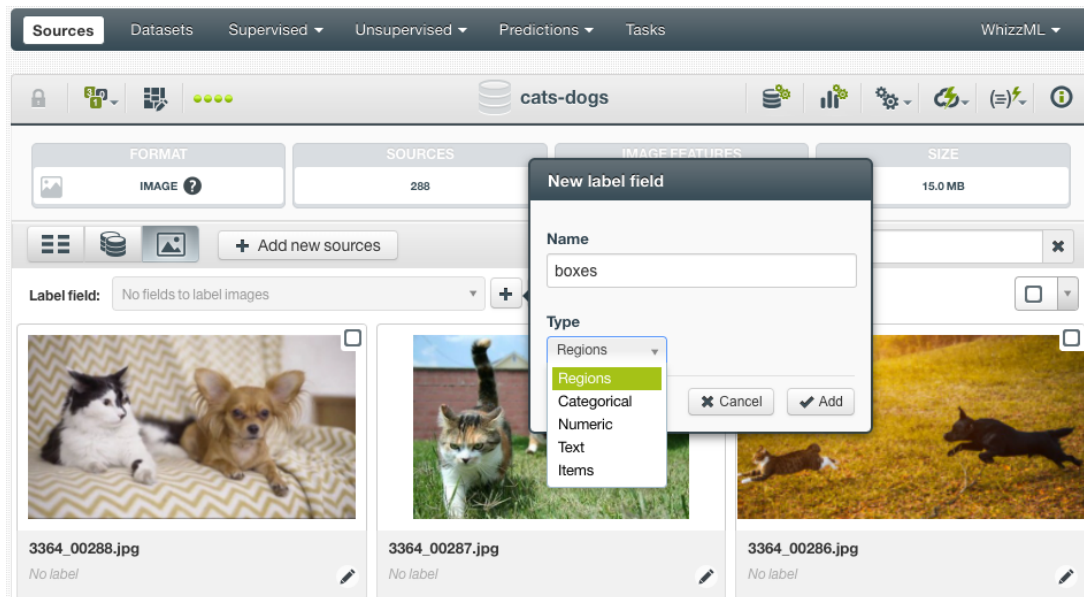


Figure 3.4: Add a regions field

Now clicking on an image in the view will bring out the annotation interface, as shown in Figure 3.5.

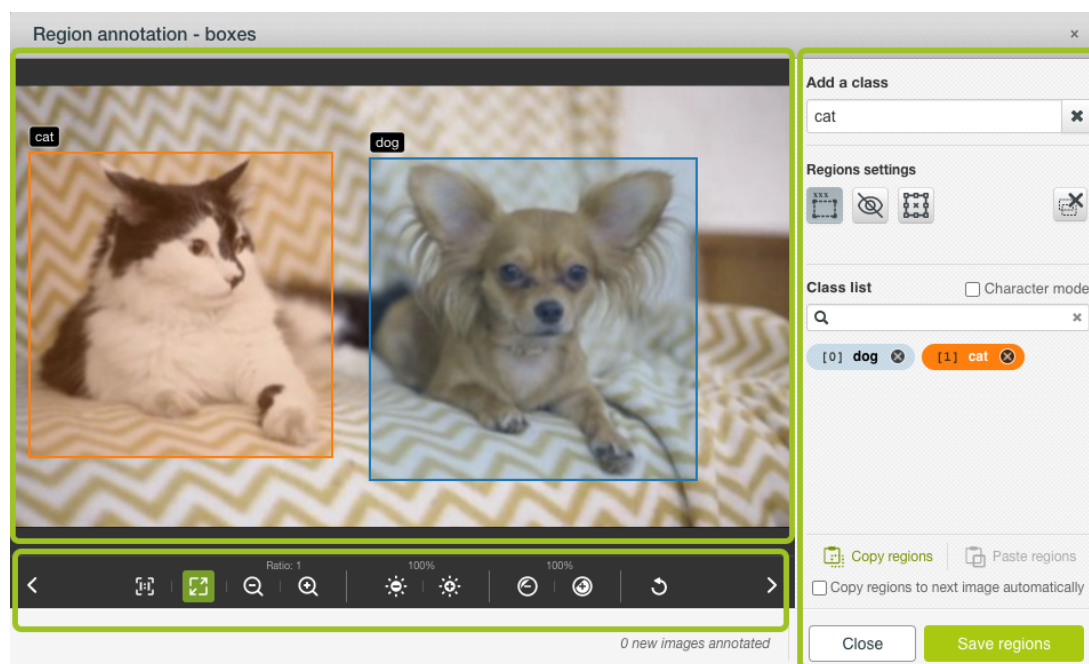


Figure 3.5: The annotation interface

The interface consists of three parts:

- **Image Panel:** The image panel is where the image is shown and is used for drawing bounding boxes.
- **Adjustment Panel:** The adjustment panel sits below the image and is made of buttons to make viewing adjustments to the image. The adjustments may help users see the image better so that it's easier to annotate the image. Note that the adjustments are for viewing only, they do not modify the image. There are six groups of controls in this panel. From left to right:

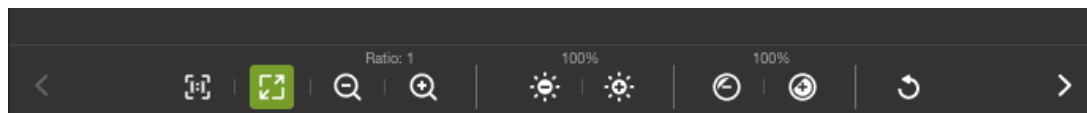


Figure 3.6: The adjustment panel

1. **Scrolling Arrows:** Use the scrolling arrows on the left and on the right to go to the next or the previous image in the composite source.
  2. **Image Size:** Click on **Actual size** icon to show the image in its actual size, or use SHIFT+G as the shortcut. Click on **Fit window** icon to fit the image to the Image Panel, or use SHIFT+F as the shortcut.
  3. **Zoom:** Click on the icon with - to zoom out on the image, or use SHIFT+O as the shortcut. Click on the icon with + to zoom in on the image, shortcut SHIFT+I. The **Ratio** on top of the two icons shows the current zoom level.
  4. **Brightness:** Click on the icon with - to decrease the brightness of the image, or use SHIFT+A as the shortcut. Click on the icon with + to increase the brightness, or use SHIFT+S as the shortcut. The current brightness is shown on top of the two button.
  5. **Contrast:** Click on the icon with - to decrease the contrast of the image, or use SHIFT+X as the shortcut. Click on the icon with + to increase the contrast, shortcut SHIFT+C. The current contrast is shown on top of the two button.
  6. **Reset:** Click on the reset icon to reset everything.
- **Control Panel:** The control panel is on the right and contains annotation controls. These controls play different roles in the annotation process. There are six sections in this Panel.

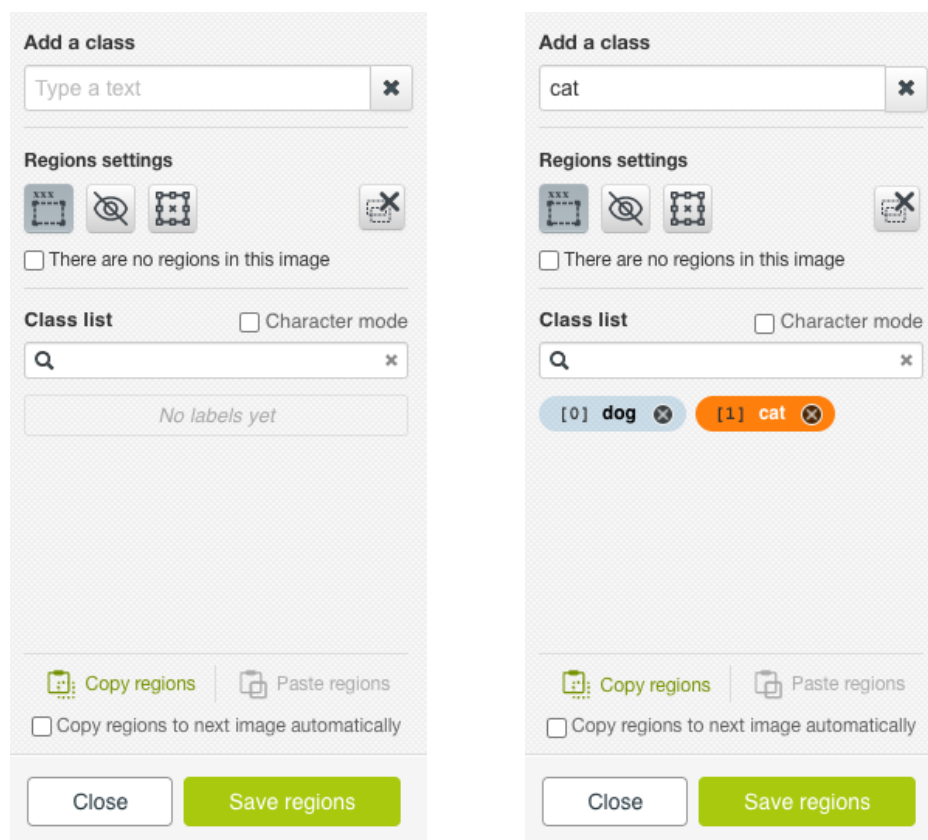


Figure 3.7: An empty Control Panel and a two-class Control Panel

1. Under **Add a class** is a text input box for entering the name of a new class. Hit **return** key to

create the class. It will then be listed under **Class list** section. Click the **x** icon to clear the input box.

2. Under **Regions settings** are four icons. They can be clicked for toggling settings of regions in the image. From left to right:
  - (a) **Show/Hide region labels** can be selected or deselected for showing or hiding regions labels, with SHIFT+L as its shortcut. The icon is selected by default. Deselecting it is useful if sometimes users only want to see the bounding boxes, not the class labels.
  - (b) **Hide/Show all regions** can be selected or deselected for hiding or showing all regions. with SHIFT+H the shortcut. This is useful when users want to drawn on the image without the interference of other regions. Once this icon is selected, users can use the class icons in the **Class list** to show or hide one or more classes, not necessarily all classes.
  - (c) **Enable/Disable region editing** can be selected or deselected for enabling or disabling the editing of regions, with its shortcut SHIFT+E. The icon is deselected by default, so regions in the images are not editable. Once it's selected, all regions in the image can be modified. Users can change their sizes, move them or delete them.
  - (d) **Remove all regions** can be clicked to remove all existing regions, and its shortcut is CTRL+SHIFT+D.
3. There is a checkbox **There are no regions in this image**, and its shortcut is CTRL+SHIFT+E. When it's checked, as shown in [Figure 3.8](#), it means there is not a single object of interest in this image, which is useful information and the image will be included in model training. As soon as a bounding box gets drawn in the image, this checkbox is automatically deselected.

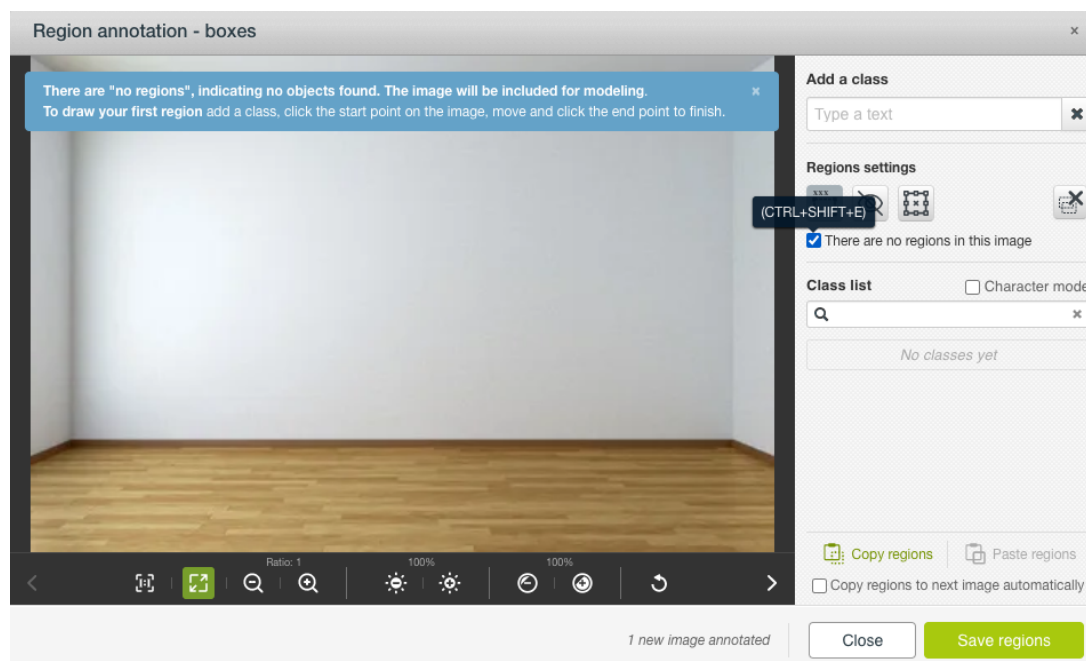


Figure 3.8: The checkbox for marking no regions in the image

4. **Class list** section lists all classes. When one class is selected, its color becomes highlighted and the bounding boxes drawn in the image will have the class as their labels. Only one class can be selected at a given time. Click on **x** to delete a class. **Character mode** is a checkbox. When checked, it adds 36 classes to the list, 0 to 9 and A to Z. It also designates the number and alphabet keys as the shortcuts to their corresponding classes. That is, press 0 then it selects 0 as the class for annotation, press v then it selects V as the class. This would make the annotation process very fast in certain situations. When the character mode is enabled, any existing classes are kept, as shown in [Figure 3.9](#). Their shortcuts are double-keys. For

instance, press 1 twice quickly enough to select *dog* class, press 1 and 2 quickly enough to select *cat* class. A search box for classes is below the section title **Class list**. Type a class name to search and filter classes. This is useful when the class list is long.

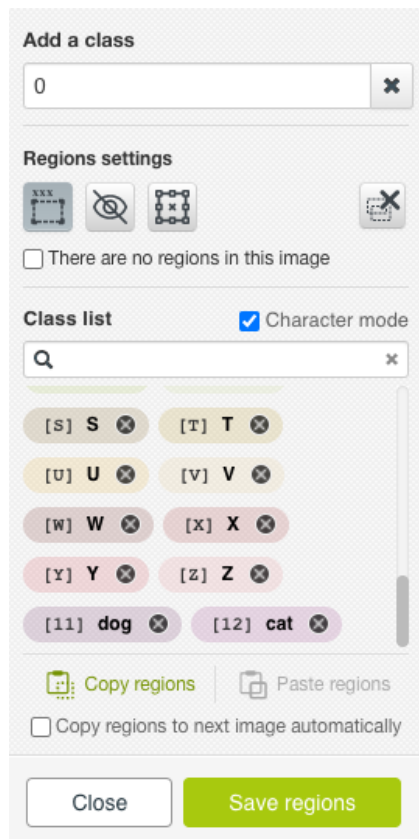


Figure 3.9: The character mode

5. **Copy/Paste** section contains icons for doing copy and paste. Note that this is for copying and pasting between images. Users can copy a region and paste it to a different image, but not in the same image. The purpose of this section is to facilitate the annotation process when many images contain similar objects, such as video frames. If the checkbox **Copy regions to next image automatically** is selected, regions in this image will be copied to the next image automatically when scrolling to it.
6. **Close** button is for closing the annotation interface. Please note that closing the interface will lose any classes or regions that are not saved. **Save regions** button is for saving all regions in the image.

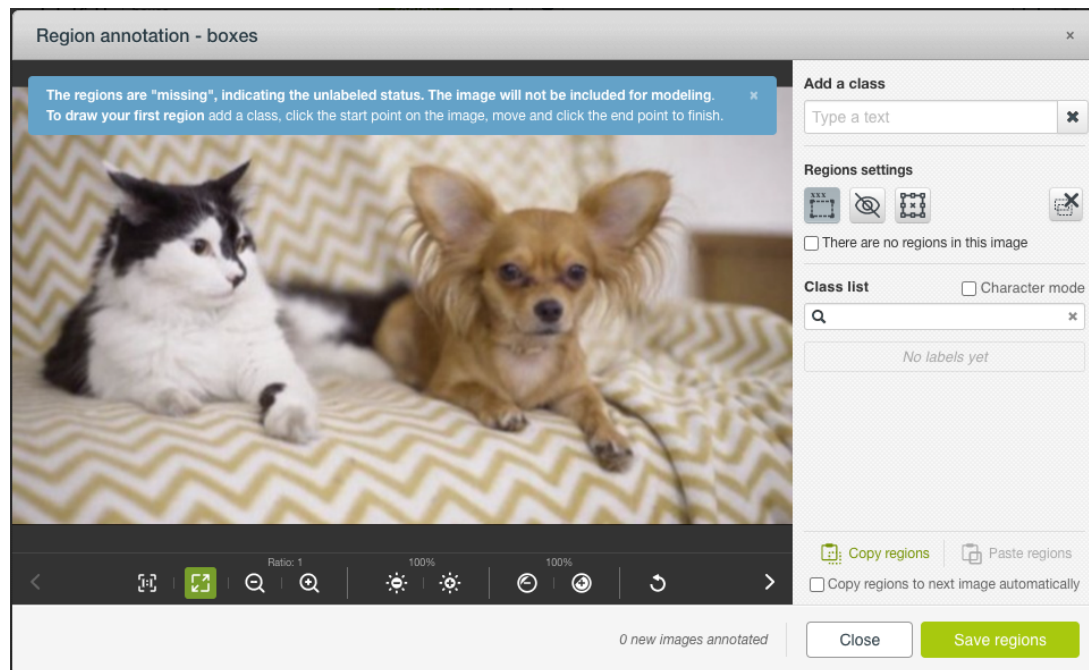


Figure 3.10: Warning for an unannotated status

**Note:** When an image is not annotated, the regions field for the image has a value of *missing*, indicating the unannotated status. The image will not be included for model training. This is the warning message when opening the annotation interface on any unannotated image, as shown in [Figure 3.10](#). But if the image does not contain any objects of interest, users should select the checkbox *There are no regions in this image*, as shown in [Figure 3.8](#). Then the regions field of the image has a value of *no regions*, indicating no objects of interest are found, and the image will be included for model training.

### 3.4 Annotating Images Interactively

We follow an example to show how to annotate images interactively on the BigML Dashboard using the annotation interface. The image composite source in the example is *cats-dogs* which has two classes, *cat* and *dog* in its images.

After uploading the images to the BigML Dashboard, an image composite was created.

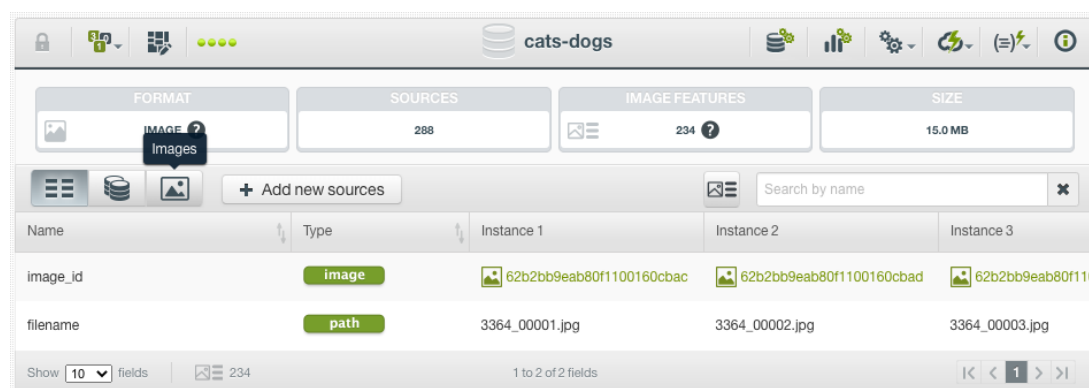


Figure 3.11: An image composite source

As shown in [Figure 3.11](#), the composite source has image and path fields, but no regions field. Go to its **Images** view, which is the third tab below the source property row (FORMAT, SOURCES, SIZE, etc.)



There is a **Label field** drop-down list. Click on the plus sign **+** to add a new label field. We name the new field `boxes`, and select the field type *Regions* from the drop-down list. Then click **Add** to add the new regions field.

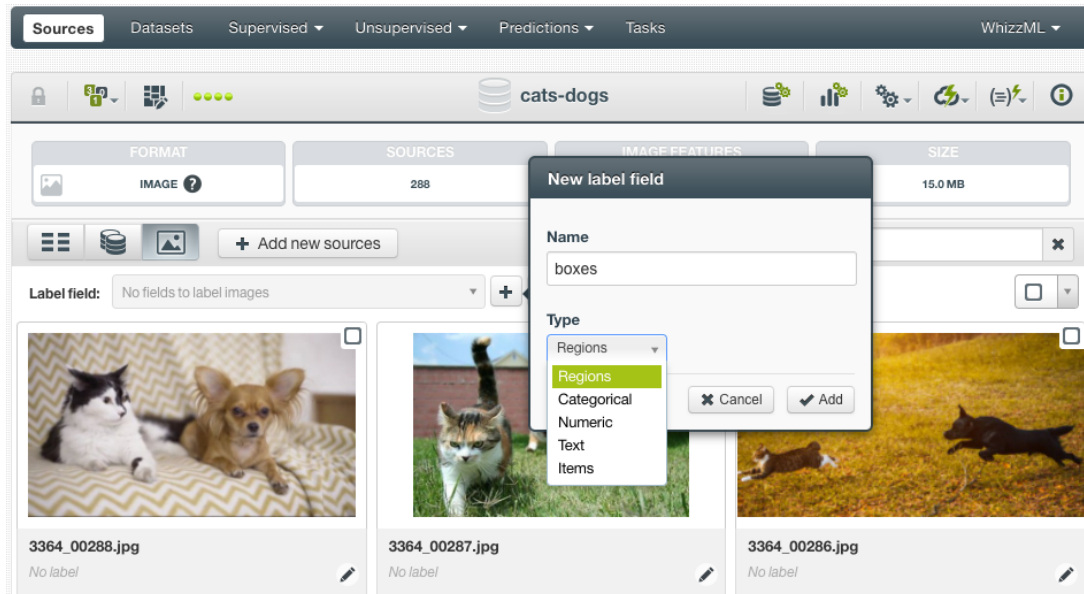


Figure 3.12: Add a regions field

Now clicking on an image will bring out the annotation interface, as shown in [Figure 3.13](#).

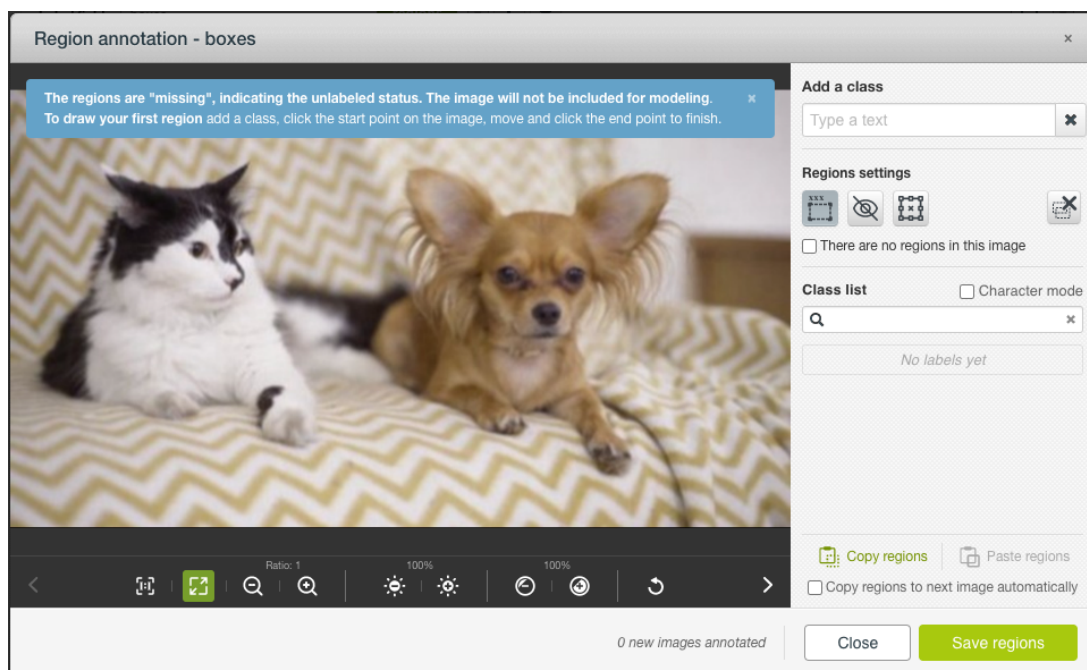


Figure 3.13: The annotation interface

We create two classes, `cat` and `dog`, by using the class input box in the Control Panel.

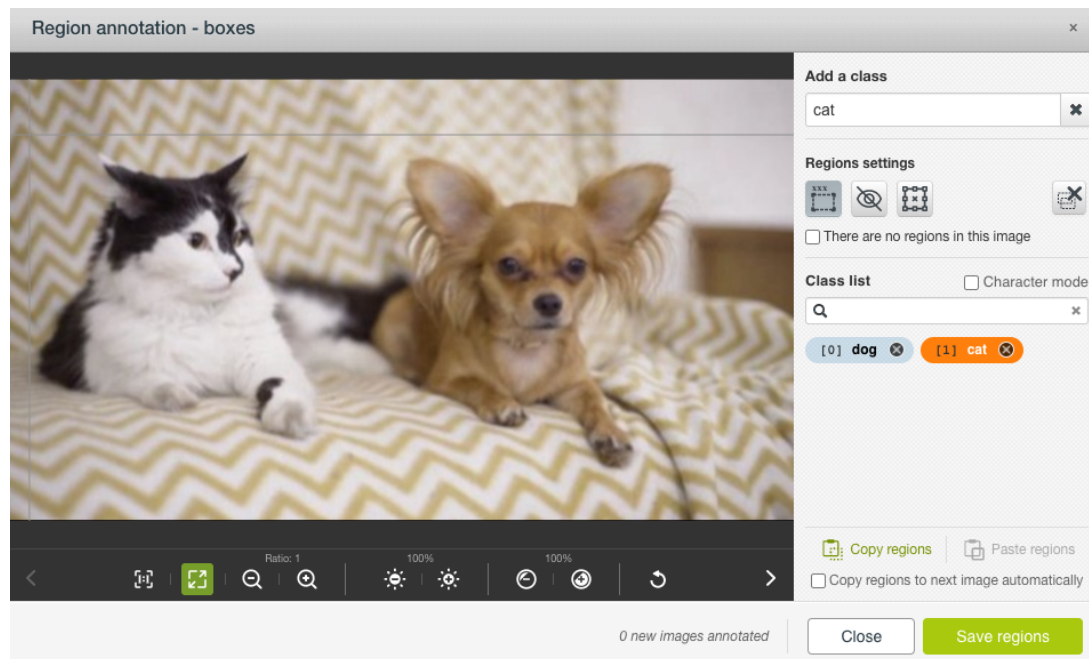


Figure 3.14: Starting drawing a bounding box

Select one class, and then draw a rectangular box to surround an object in the image. We draw the box by first clicking a point in the image as a vertex of the rectangle, and then clicking another point as its opposing diagonal vertex. When clicking the first point, the cross-hairs are very useful for estimating the coverage of the rectangle.

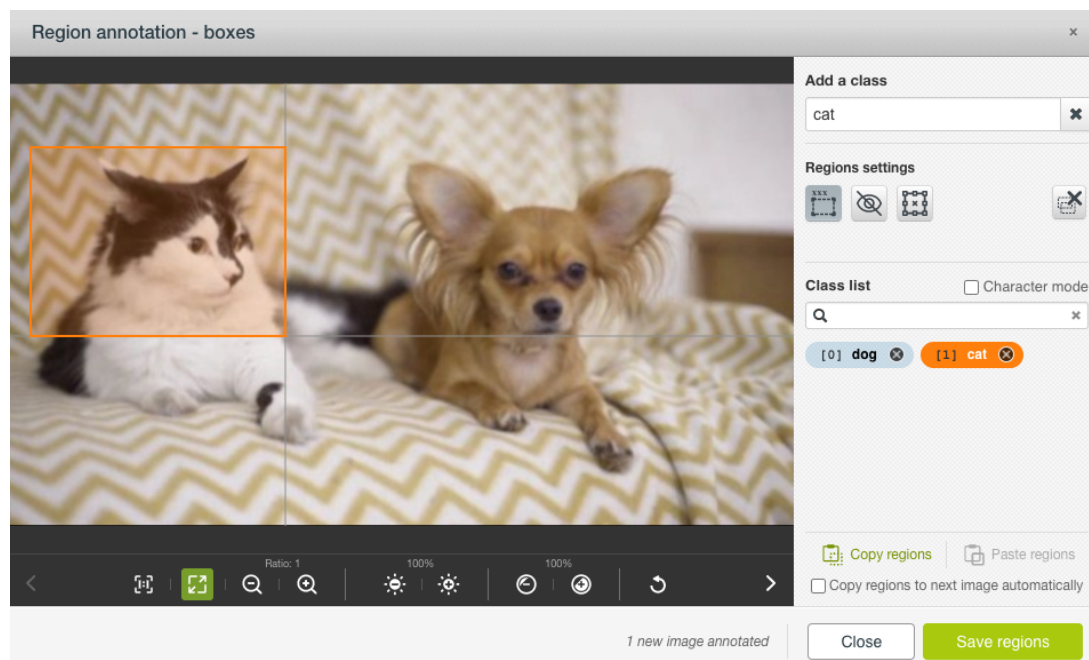


Figure 3.15: Finished drawing a bounding box

After finishing the drawing, it also labels the bounding box as the class we selected. This finishes the annotation of one region.

We can edit the region by clicking on the editing icon in the Control Panel or use the shortcut SHIFT+E. We can move the box or change its size, or delete region and start over.



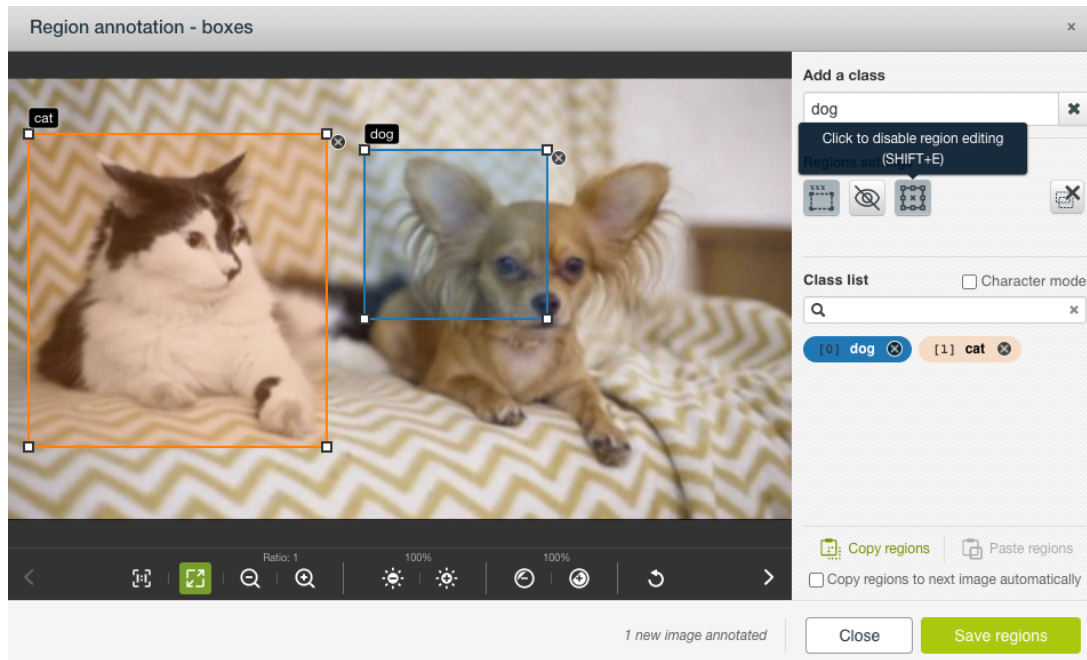


Figure 3.16: Editing regions

There are options and settings to make the annotation easier and quicker, as explained in the previous section. For instance, when there are many objects in the image, their bounding boxes may overlap each other. In such situations, it's better to hide class labels, or hide certain classes or all classes so that it's easier to see an object when drawing. These options are available in the regions setting section of the Control Panel, as described earlier. We can also copy regions to the next image, either manually or automatically.

Once we finish drawing regions on all objects in an image, we click on **Save regions** to save the annotations. Then we can go to the next image for more annotation by clicking on the scrolling arrow in the Adjustment Panel.

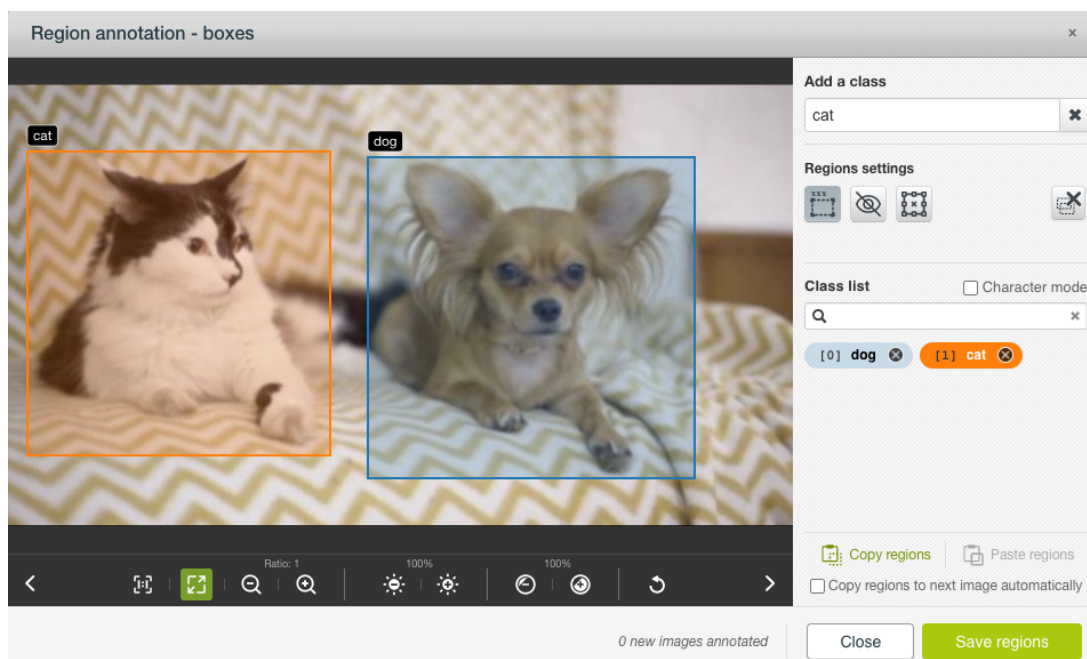


Figure 3.17: Annotation of an image finished

### 3.5 Cropping Images By Regions

One application of annotations in object detection is to crop images by their bounding boxes. This is useful in some workflows where after certain objects of general classes are detected, further machine learning is needed to identify specifics from those objects. A good example would be reading license plates on automobiles. After a model detects the license plates in images, they can be cropped to images containing only the plates, which would be used for reading the license numbers.

BigML provides a *Cropping* utility. It takes an image composite source as the input, crops each image in the composite by its regions, and create a new image composite source which consists of cropped regions as its component sources. The number of components (or images) of the output composite source is equal to the total number of regions in the input composite source.

From a composite source with a regions field, click CROP IMAGES BY REGIONS in the **gear action menu**, as shown in Figure 3.18.

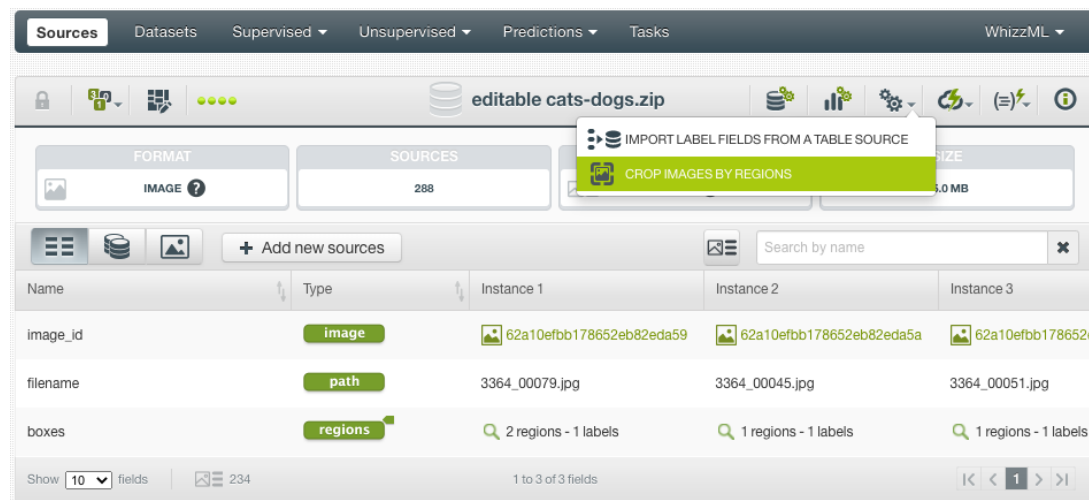


Figure 3.18: Cropping images by regions

It leads to the cropping configuration panel, as shown in Figure 3.19. There are three options to configure.

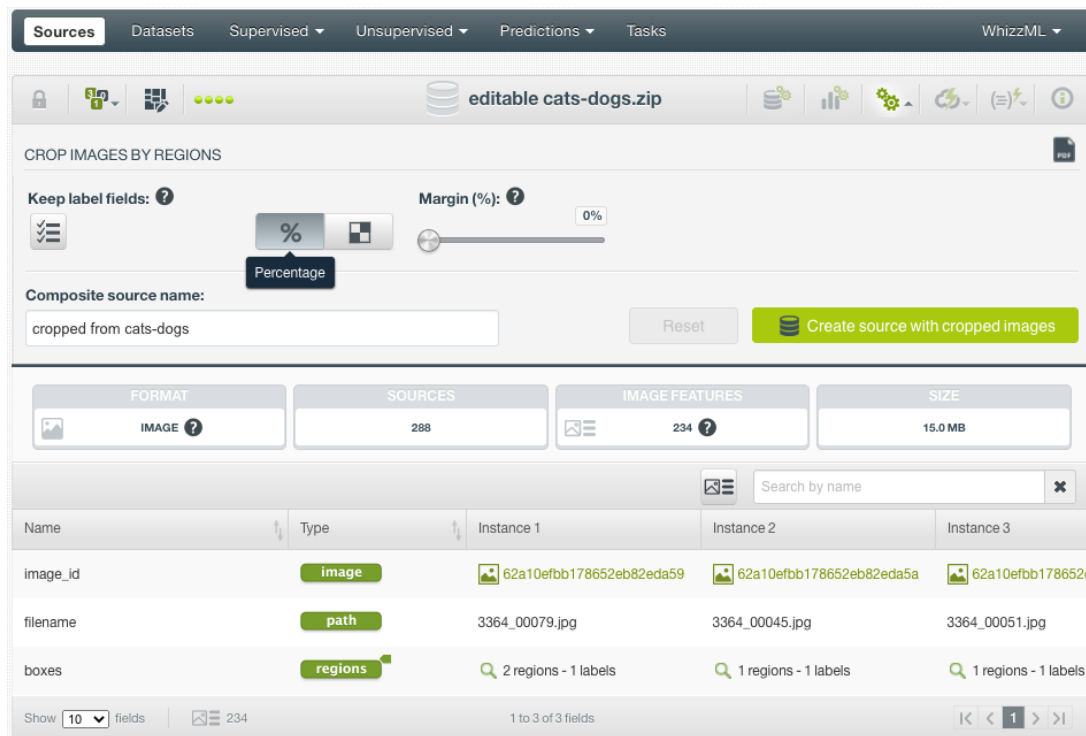


Figure 3.19: Configuration options of cropping

- **Keep label fields** is a checkbox. If it's enabled, all label fields in the input composite source will be passed to the output composite. Each cropped image will inherit those field values from the image they belong to in the input composite. By default, this option is disabled.

Note: If **Keep label fields** is enabled, the label fields added to the output composite are in addition to the label field created from the regions field of the input composite. In other words, the class label in the regions is always passed to the cropped images in the output composite. The field name takes the format of "<regions-field-name>-label". In the example of *cats-dogs*, the regions field is named *boxes*. After it's cropped, the output composite will have a label field named *boxes-label*. Let's say there is a region in the input composite, and its class label is *cat*. After cropping, the region becomes a component image in the output composite, and the *boxes-label* field of the image has a value of *cat*.

- **Margin** is used for adding margins when cropping the images. It can be specified either in pixel or in percentage. When given in pixel, it will add the number of pixels as margins to all sides of the cropped image. If given in percentage, it will add margins as percentages of the region's width and height, respectively. For instance if a region is sized 300x200 (300 pixels of width, 200 pixels of height), a 5% margin will add 15 pixels to both ends of its width, and 10 pixels to both ends of its height. By default, the margin is set to 0.
- **Composite source name** is used to input the name of the output composite source. By default it's set to "regions cropped from <input-composite-name>".

After configuring the options, click on the green button **Create source with cropped images** to perform the cropping.

After the cropping is finished, a new image composite source is created. Go to its **Images** view to view the cropped images.

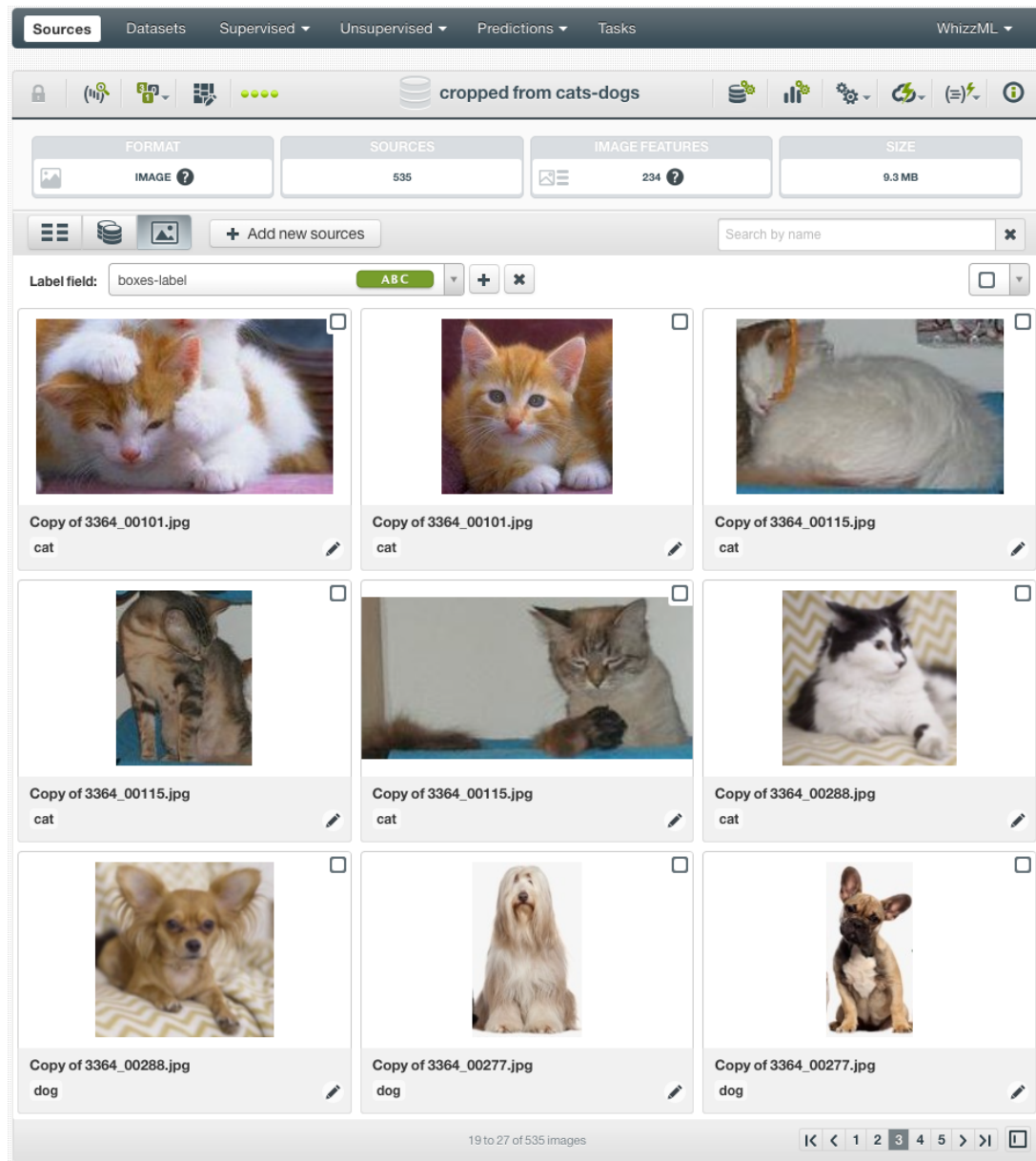
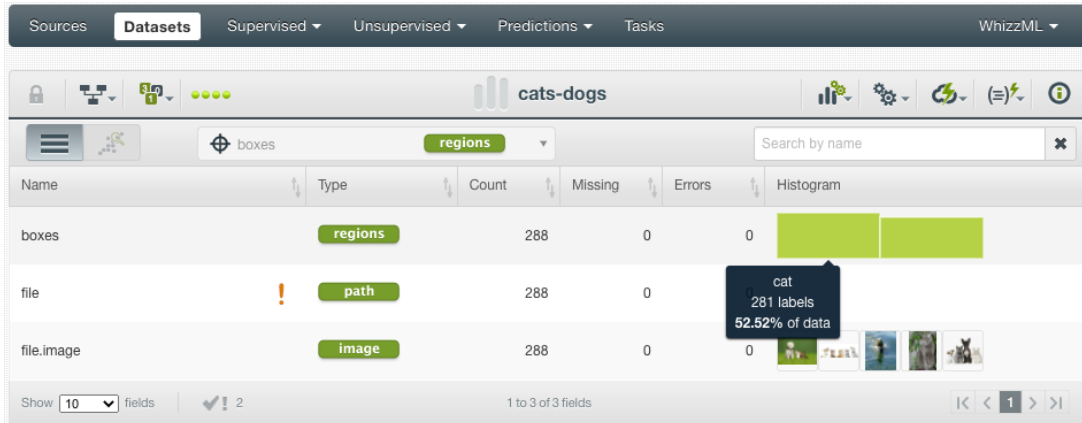


Figure 3.20: Cropped images in the output composite source

## Creating Object Detection Deepnets

A dataset can be created from a composite source. When the composite source has a regions field, it's automatically set as the objective field.



Name	Type	Count	Missing	Errors	Histogram
boxes	regions	288	0	0	
file	path	288	0	0	
file.image	image	288	0	0	

Figure 4.1: Dataset with a regions field

For more on datasets, including their creation and options, please refer to section Creating Datasets of the [Datasets with the BigML Dashboard](https://static.bigml.com/pdf/BigML_Datasets.pdf)<sup>1</sup>[4].

### 4.1 Creating Object Detection Deepnets with 1-Click

To create an object detection deepnet, first make sure the objective field is a regions field.

The 1-CLICK DEEPNET option is in the **1-click action menu** from the dataset view. See [Figure 4.2](#).

<sup>1</sup>[https://static.bigml.com/pdf/BigML\\_Datasets.pdf](https://static.bigml.com/pdf/BigML_Datasets.pdf)

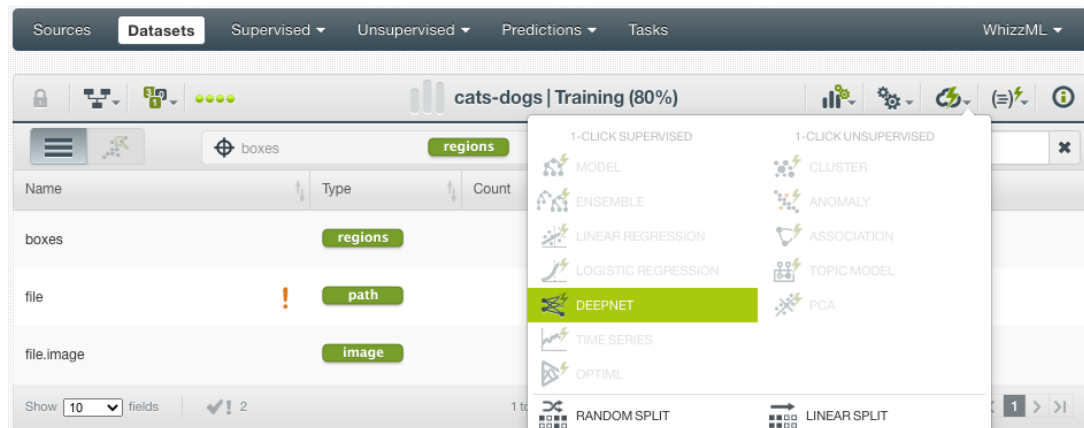


Figure 4.2: Create deepnet from 1-click action menu

Alternatively, use 1-CLICK DEEPNET option in the **pop up menu** from the dataset list view. (See [Figure 4.3](#).)

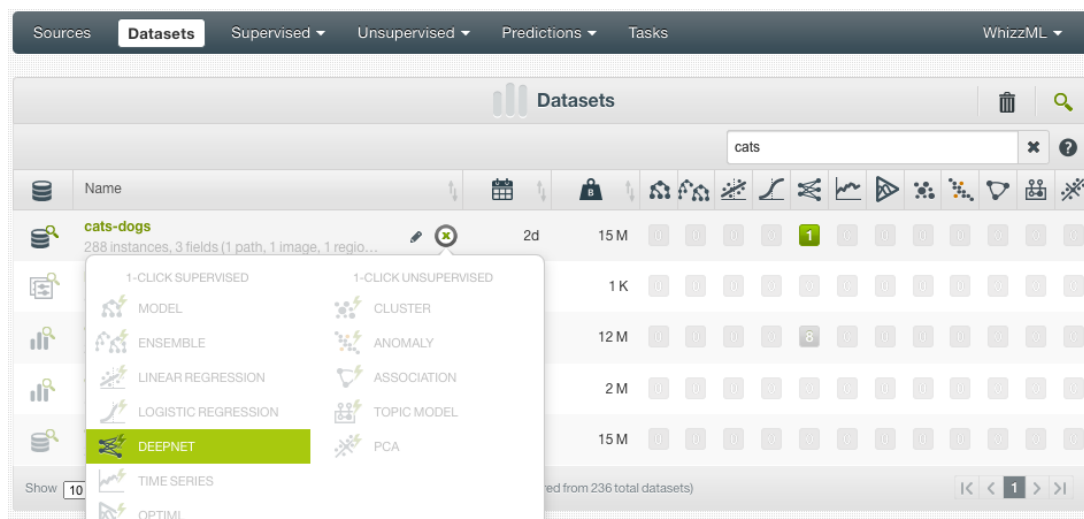


Figure 4.3: Create deepnet from popup menu

With a regions field as the objective field, the object detection deepnet trained will be a convolutional neural network (CNN). Please refer to section Understanding Deepnets of the [Classification and Regression with the BigML Dashboard](#)<sup>2</sup>[2] for more information.

**Note:** Object detection deepnets are trained from raw image pixels. If their datasets contain extracted image feature fields, they will be ignored during training.

## 4.2 Object Detection Deepnet Configuration Options

BigML provides a number of parameters that users can configure when creating an object detection deepnet. They are very similar to those of other deepnets.

Click on the CONFIGURE DEEPNET menu option in the **configuration menu** from the dataset view. See [Figure 4.4](#).

<sup>2</sup>[https://static.bigml.com/pdf/BigML\\_Classification\\_and\\_Regression.pdf](https://static.bigml.com/pdf/BigML_Classification_and_Regression.pdf)

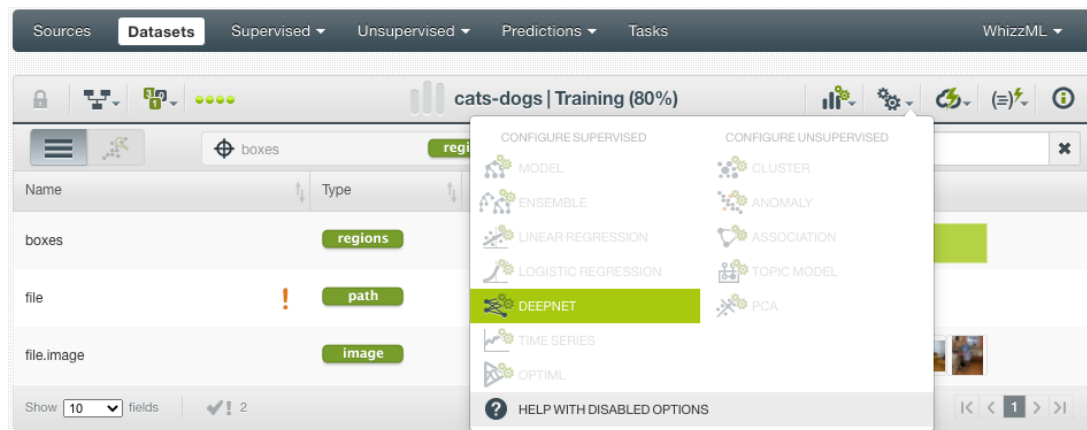


Figure 4.4: Create deepnet by configuring options

Then comes the deepnet configuration panel, as shown in Figure 4.5.

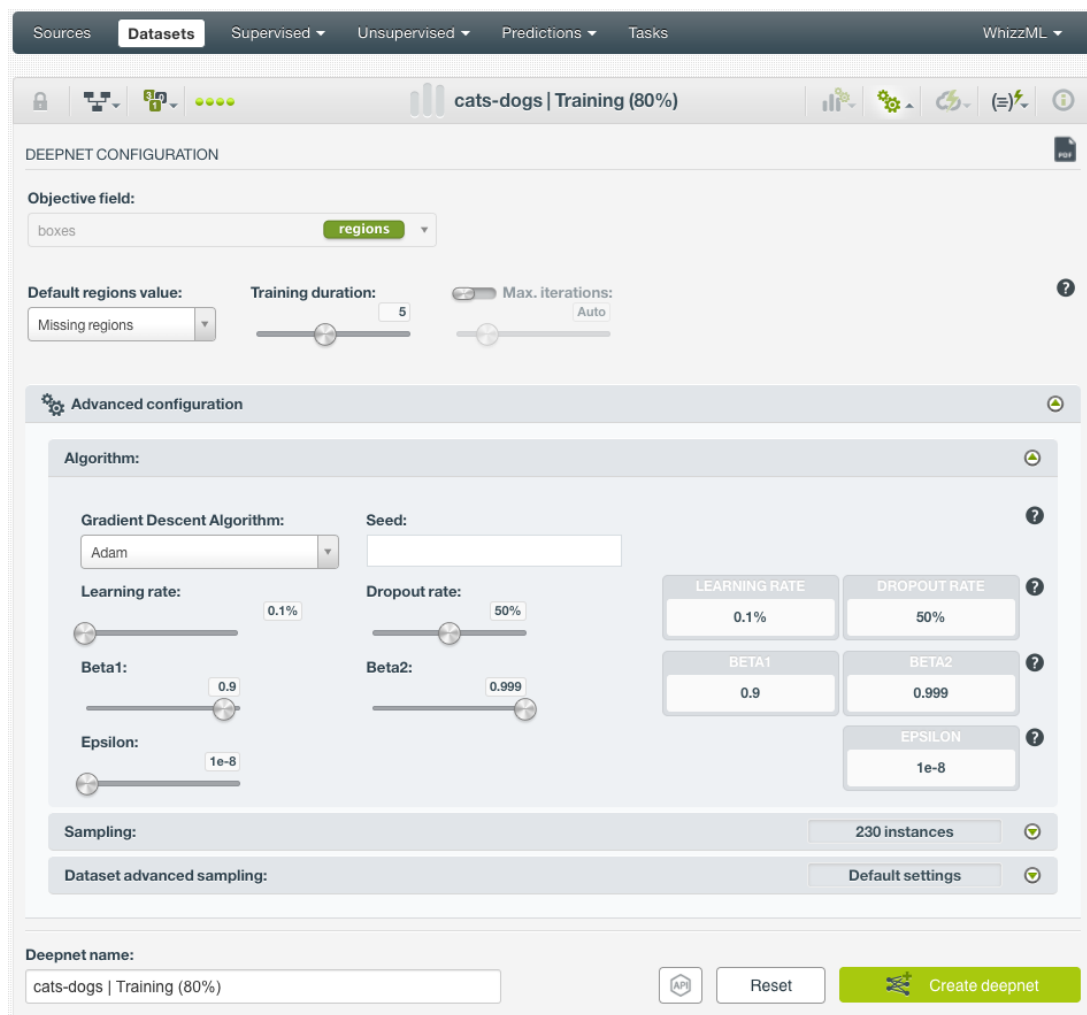


Figure 4.5: Deepnet configuration options

### 4.2.1 Objective Fields

The objective fields are the fields you want to predict. Object detection only supports **regions** fields as **objective fields**.



## 4.2.2 Default Regions Value

If there are unannotated images in the dataset, the regions field of these images will have a value of *Missing regions* and they won't be included for training. You can change this by selecting *No regions* for this option. Then all unannotated images will have a value of *No regions*, which means there are no objects of interest in the images, and they will be included for training. See [Figure 4.6](#).

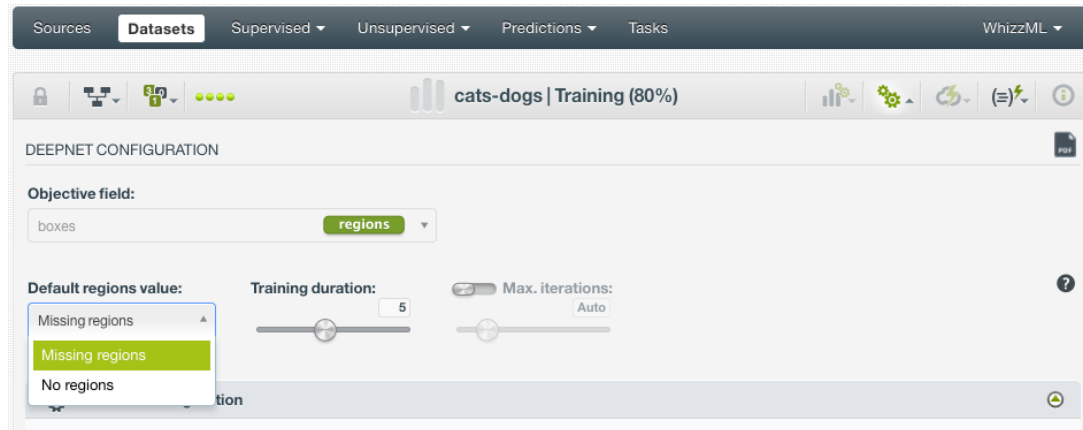


Figure 4.6: Default regions value

## 4.2.3 Training Duration

This is the scale parameter to regulate the deepnet training time. It's set as an integer from 1 to 10. It indicates the user preference for the amount of time they wish the optimization to take. The higher the number, the more time that users are willing to wait for possibly better deepnet performance. The lower the number, the faster that users wish the deepnet training to finish. The default value is set to 5. See [Figure 4.7](#).

The training duration is set in a scale. The actual training time depends on the dataset size, among other factors.

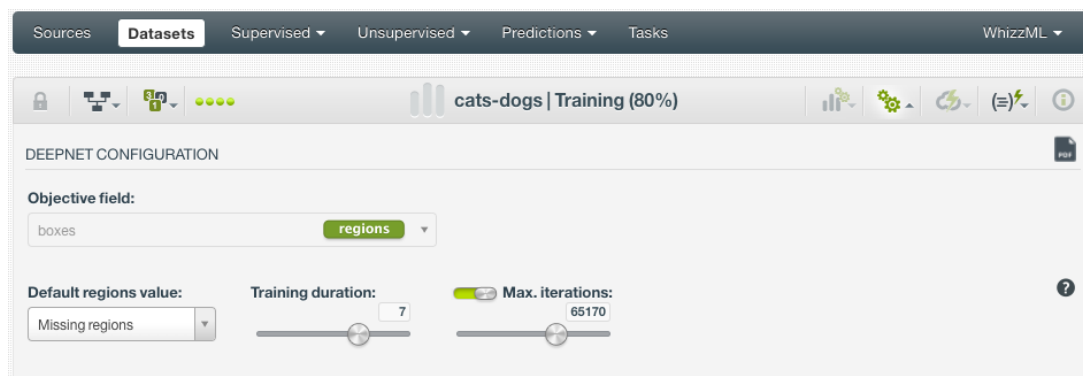


Figure 4.7: Set the training duration for a deepnet

## 4.2.4 Maximum Iterations

The number of iterations in a deepnet is the **number of gradient steps** the algorithm takes during the optimization process. You can set the maximum number of iterations to train the object detection deepnet by activating this option using the switch, as shown in [Figure 4.7](#). By default, this option is deactivated, in which case BigML will stop training the deepnet if a certain number of iterations goes by without substantial progress or if it reaches the training duration limit (see [Subsection 4.2.3](#)).

The rest of the configuration options, such as Algorithm and Sampling, are exactly the same as in other deepnets. Please refer to section Deepnets Configuration Options of the [Classification and Regression](#)



with the BigML Dashboard<sup>3</sup>[2] for details.

---

<sup>3</sup>[https://static.bigml.com/pdf/BigML\\_Classification\\_and\\_Regression.pdf](https://static.bigml.com/pdf/BigML_Classification_and_Regression.pdf)

# Visualizing Object Detection Deepnets

## 5.1 Introduction

BigML provides object detection visualization so that you can analyze and interpret the results. After an object detection deepnet is created, users are presented with the Object Detection Visualization Page, which is shown in [Figure 5.1](#).

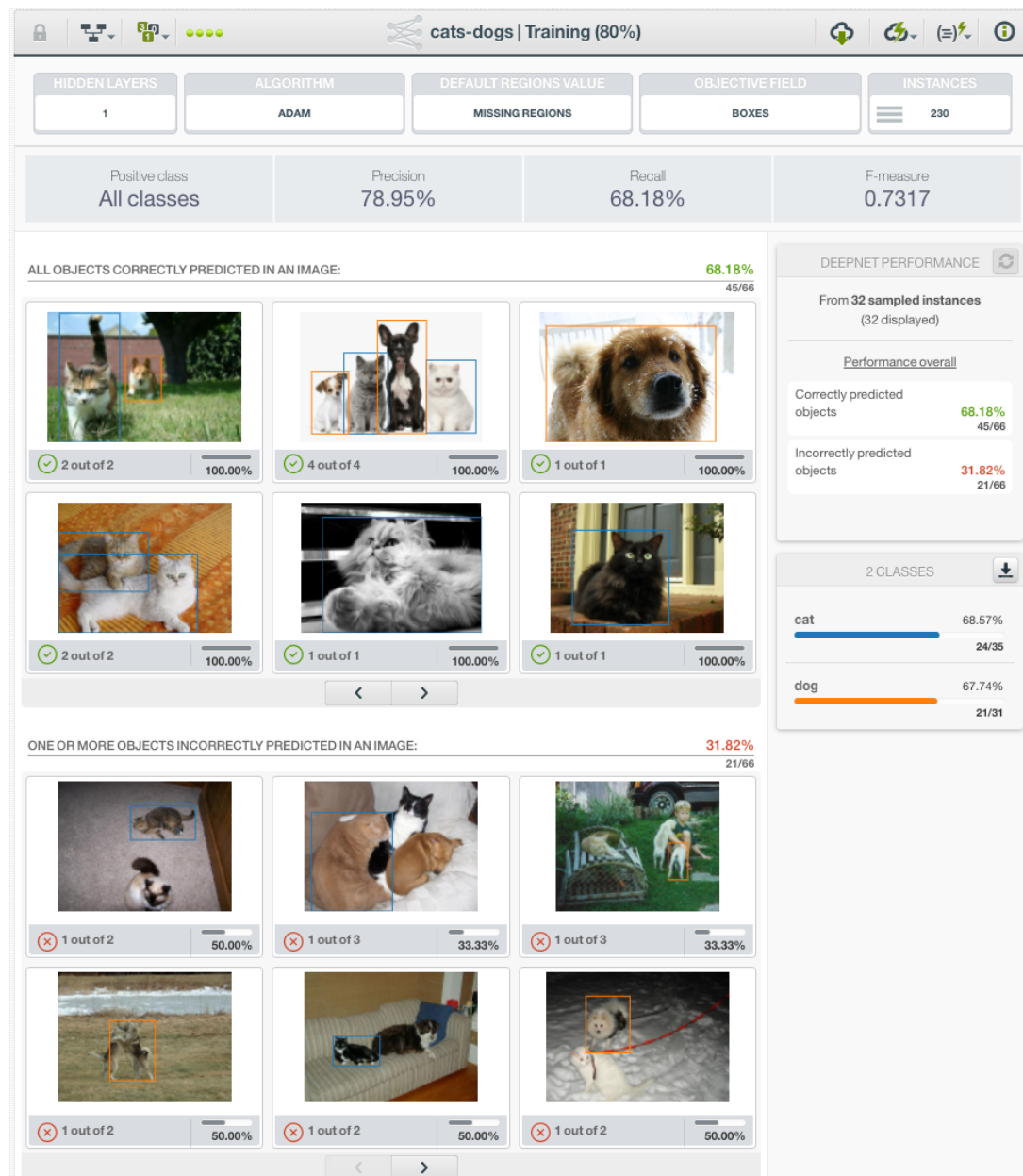


Figure 5.1: Object detection visualization page

On the top row of the page are deepnet parameters which include its hidden layer number, algorithm and the default regions value. The top row also lists the objective field of the deepnet and its number of instances.

The visualization shows the performance of the deepnet on a set of sampled instances. This set is called *holdout set*, and it is used for validation during deepnet training.

Below the top row, the page shows **Positive class**, **Precision**, **Recall** and **F-measure**. Positive class is the class of objects used in the metrics that's being shown below the top two rows. It could be all classes or a specific class. When it's all classes, the metrics below are based on all objects. If it's a specific class, the metrics below are derived from the objects of that class.

## 5.2 Visualization Page Layout

In addition to the top two rows of parameters and metrics, the visualization page has three main sections: the **Image Results**, the **Performance Panel** and the **Class List**. See Figure 5.2.

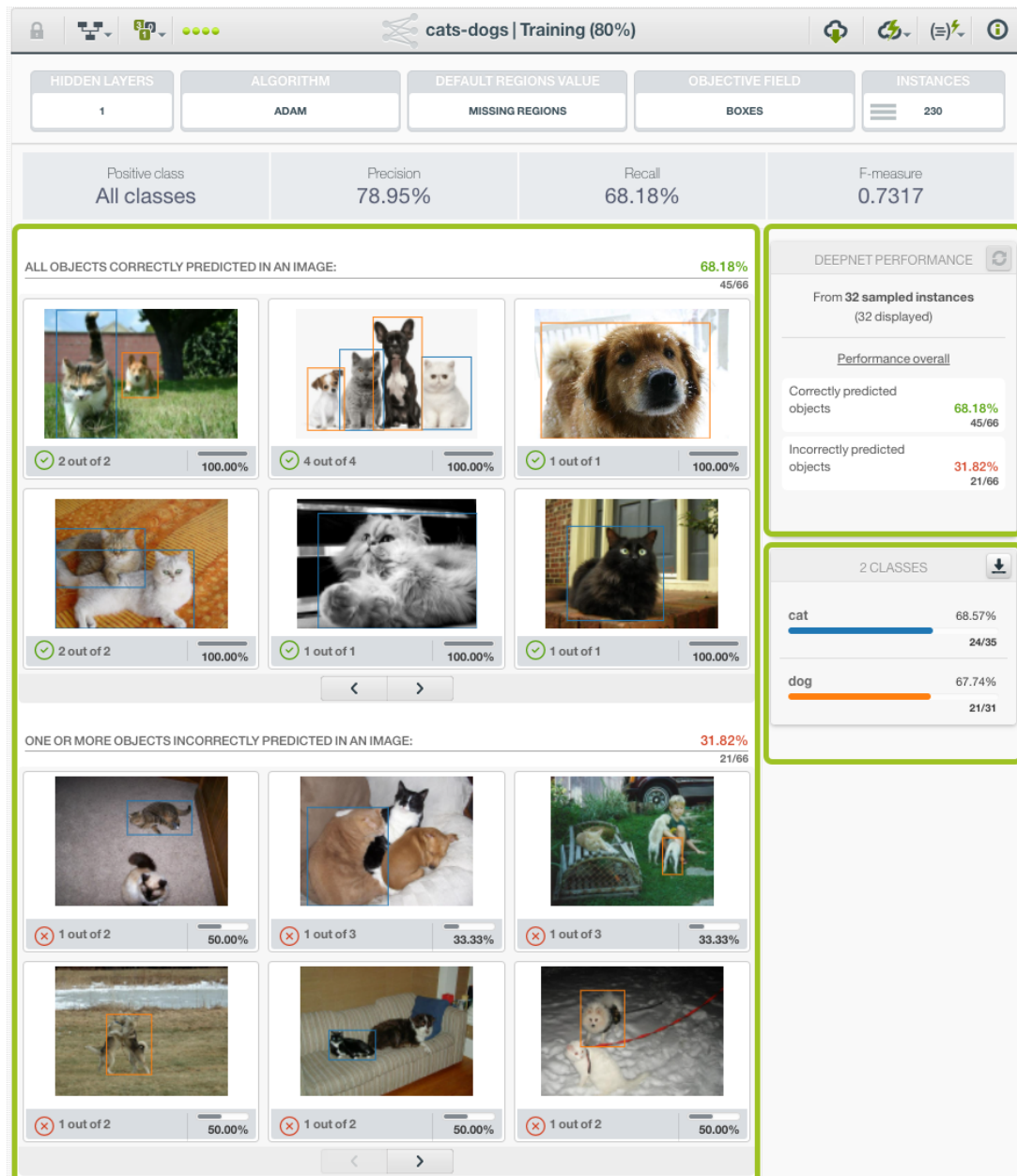


Figure 5.2: Object detection visualization layout

### 5.2.1 Performance Panel

The **Performance Panel** shows the performance metrics of the object detection deepnet on the holdout set. See Figure 5.3. The number of instances of the holdout set is displayed under the panel heading. The performance panel shows the overall performance for all classes in the holdout set by default. However when a class is selected in the Class List, the Performance Panel shows the performance for that specific class.

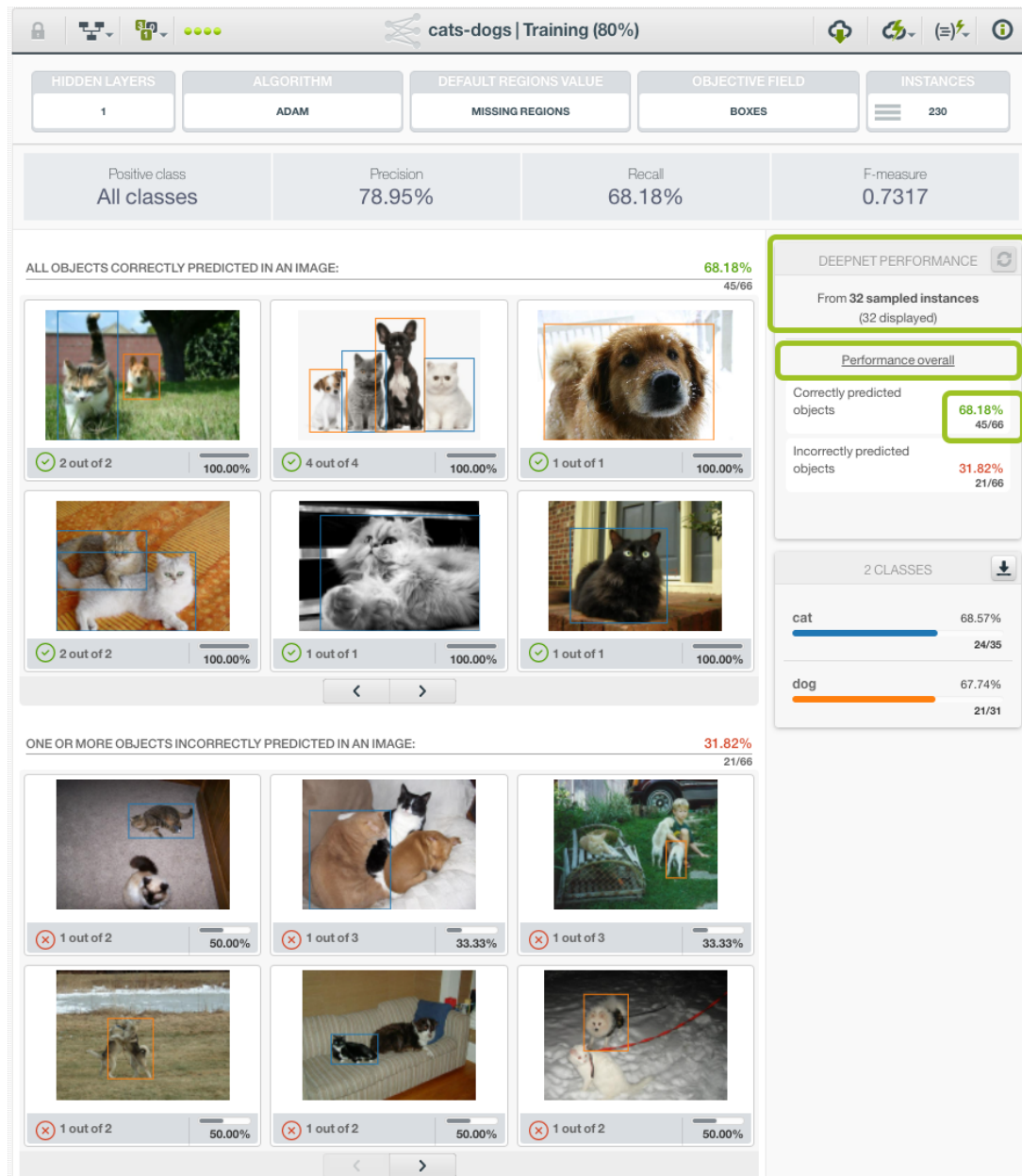


Figure 5.3: Object detection visualization performance panel

The performance metrics, either correctly predicted or incorrectly predicted, are provided as both percentage and count in objects. For instance, when the count is 45/66 as shown in Figure 5.3, that means there are 66 objects altogether in all images of the holdout set, and 45 of them were predicted correctly.

## 5.2.2 Image Results

The **Image Results** section shows two subsections, with each a paginable list of images from the holdout set. The upper list is titled **All Objects Correctly Predicted In An Image**, and the lower list **One or More Objects Incorrectly Predicted In An Image**. Each list shows up to six images and more images can be scrolled using the arrow buttons below the list. In both lists, every image has a caption. At its left end, the caption shows the recall in the form of  $m$  out of  $n$  where  $m$  is the number of correctly predicted regions (we use objects and regions interchangeably) in the image,  $n$  the number of ground truth regions in the image. At its right end the caption shows the image's recall in percentage. In the upper subsection, because all objects in each image are predicted correctly, all images have 100% recall. In the lower subsection, there are incorrectly predicted objects in every image. At the right end of a caption, the length of the solid color in the recall bar is proportional to its value. See Figure 5.4.

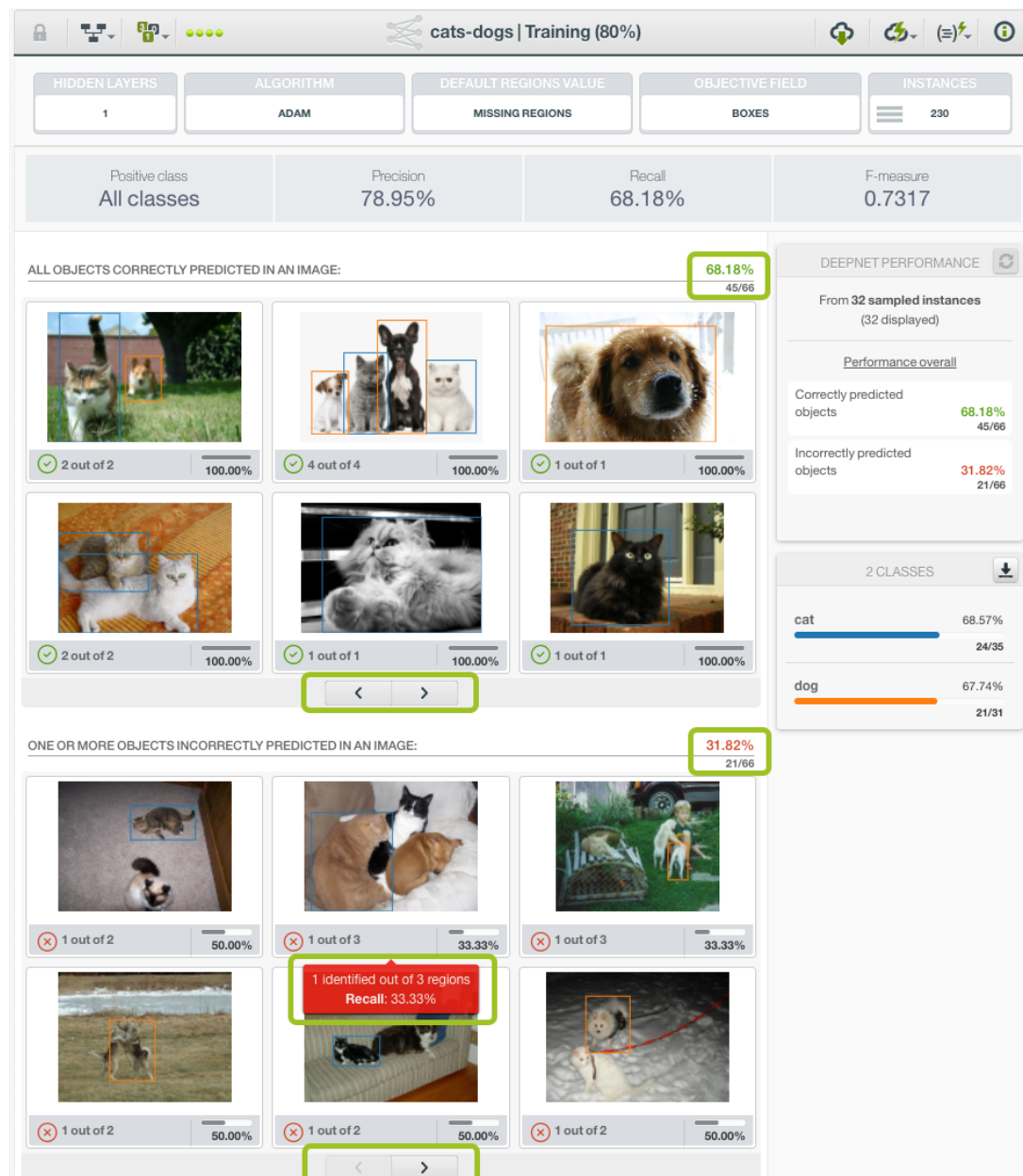


Figure 5.4: Object detection visualization image results

Clicking on an image in the Image Results section will get to a closeup view of the its prediction results.

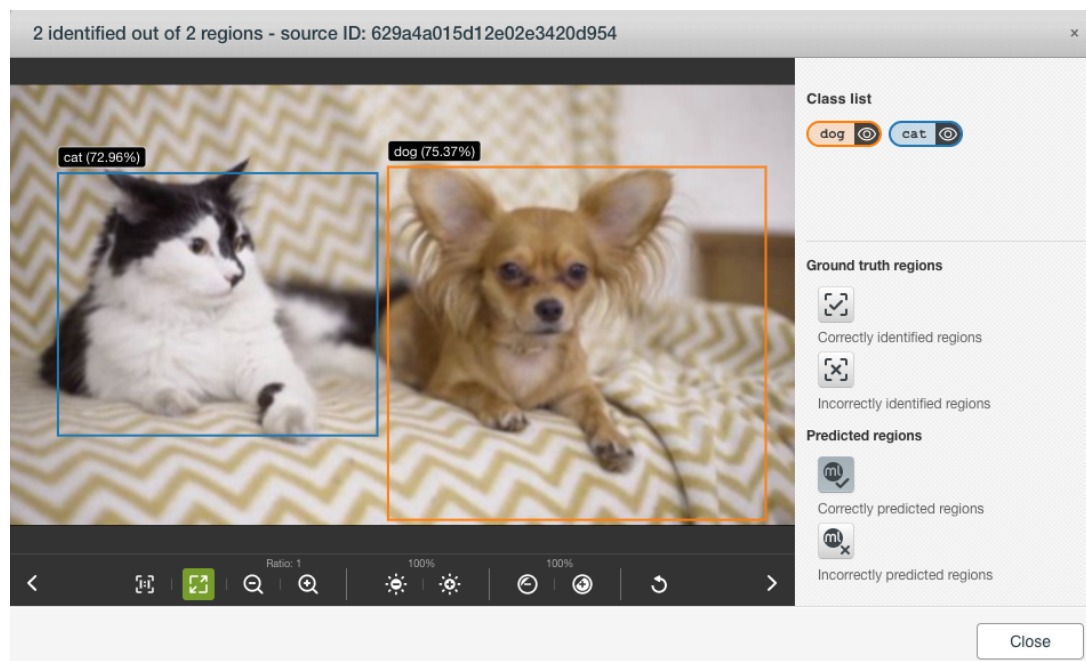


Figure 5.5: Closeup view of an image

### 5.2.2.1 Image Closeup View

In the closeup view, the image panel can show ground truth regions, as well as predicted regions along with their prediction scores. Ground truth regions are the regions that come from annotations and are supposed to surround objects accurately. Predicted regions are the regions that are predicted by the object detection deepnet.

There are two sections of viewing controls on the right. Users can use them to compare predicted results with ground truth, therefore to gauge the quality of the predictions. See [Figure 5.6](#).



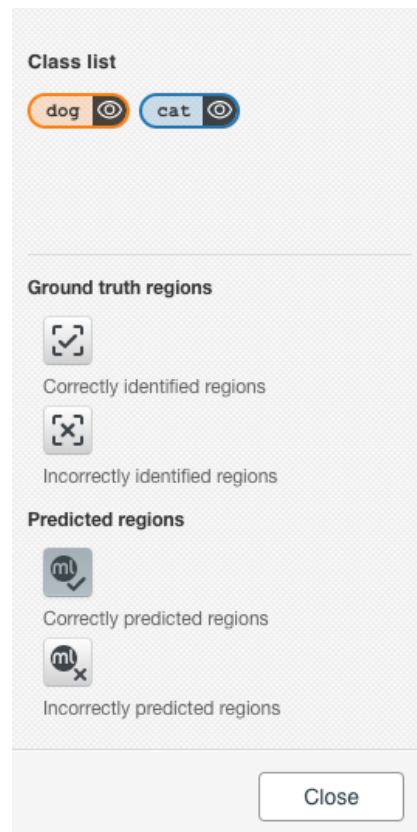


Figure 5.6: Viewing controls in the image closeup view

- On the bottom is the **Regions Control** section, which has four checkboxes. The checkboxes are selected or deselected for showing or hiding certain regions. They are divided to two groups. One group is **Ground truth regions** and the two checkboxes in it are controls of whether to show ground truth regions. If the **Correctly identified regions** checkbox is selected, the image view shows the ground truth regions that have high overlap with at least one predicted region. If it's deselected, those ground truth regions are hidden. If the **Incorrectly identified regions** checkbox is selected, the image view shows the ground truth regions that have no high overlap with any predicted regions. If both are selected, all ground truth regions are shown in the image view. If neither is selected, no ground truth regions are shown.

Another group of checkboxes is **Predicted regions**. If the **Correctly predicted regions** checkbox is selected, the image view shows the predicted regions that have high overlap with at least one ground truth region. If it's deselected, these predicted regions are hidden. When the closeup view is opened from the visualization page, only this checkbox is selected by default. If the **Incorrectly predicted regions** is selected, the image view shows the predicted regions that have no high overlap with any ground truth regions. Otherwise, they are hidden. If both checkboxes are selected, all predicted regions are shown. If both are deselected, no prediction regions are shown.

- On the top is the **Classes** section. It lists classes enabled by the lower section. The list may not include all classes from the annotations because some ground truth regions may be hidden and predicted regions may not have all classes.

Each class has a clickable eye icon. Users can click on the icon to hide or show the regions of the class. In other words, all regions of one class can be hidden or shown using the class icon. This is useful in filtering one or more classes.

### 5.2.3 Class List

The **Class List** shows all the classes in the holdout set, sorted by their number of occurrences. In other words, the list is ranked by class popularity. See Figure 5.7. The color of the class bar matches the color of the bounding boxes drawn on the objects of such class. The recall rate of each class is displayed by



percentage and count at the right side of class bar. Recall is defined as the number of correctly predicted objects for the class divided by the total number of ground truth objects for the class. The length of the color bar in proportion to the full class bar corresponds to the value of the recall of the class.

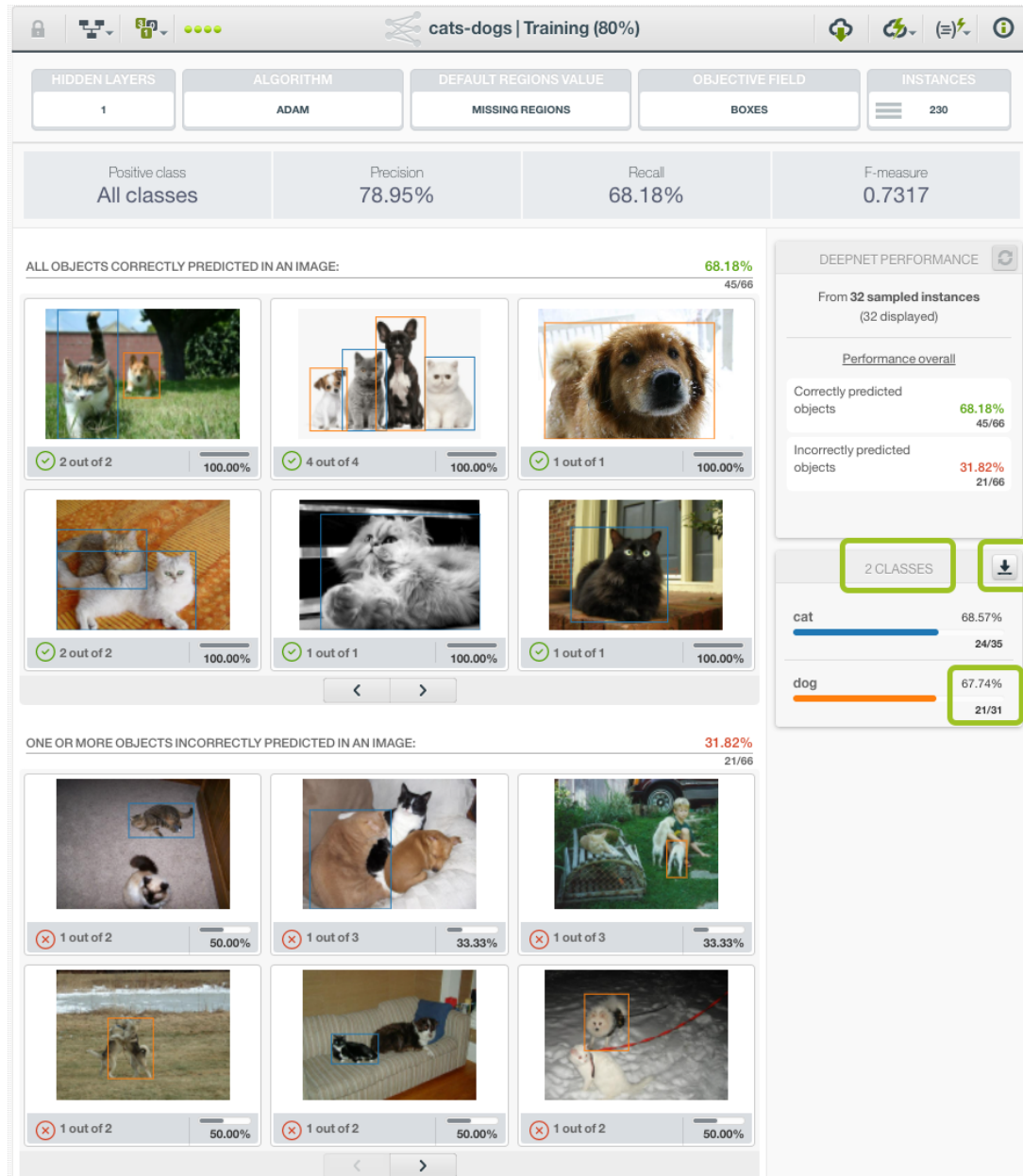


Figure 5.7: Object detection visualization class list

The Class List is scrollable when there are not enough space to show all of them together. There is also a download icon in the section heading that users can use to download the class list as a CSV.

The Class List is clickable. A class can be selected by clicking on its class bar. When this happens, the Performance Panel shows the performance metrics with respect to the class only. In addition, The Image Results section shows only images that contain objects of this class. For instance, when clicking on *cat* class in Figure 5.8, the **Correctly predicted objects** count becomes 23/32, which means there are 32 objects of *cat* in all images, and 23 of them are predicted correctly. The images that are being displayed in both subsections all contain at least one object of *cat*.

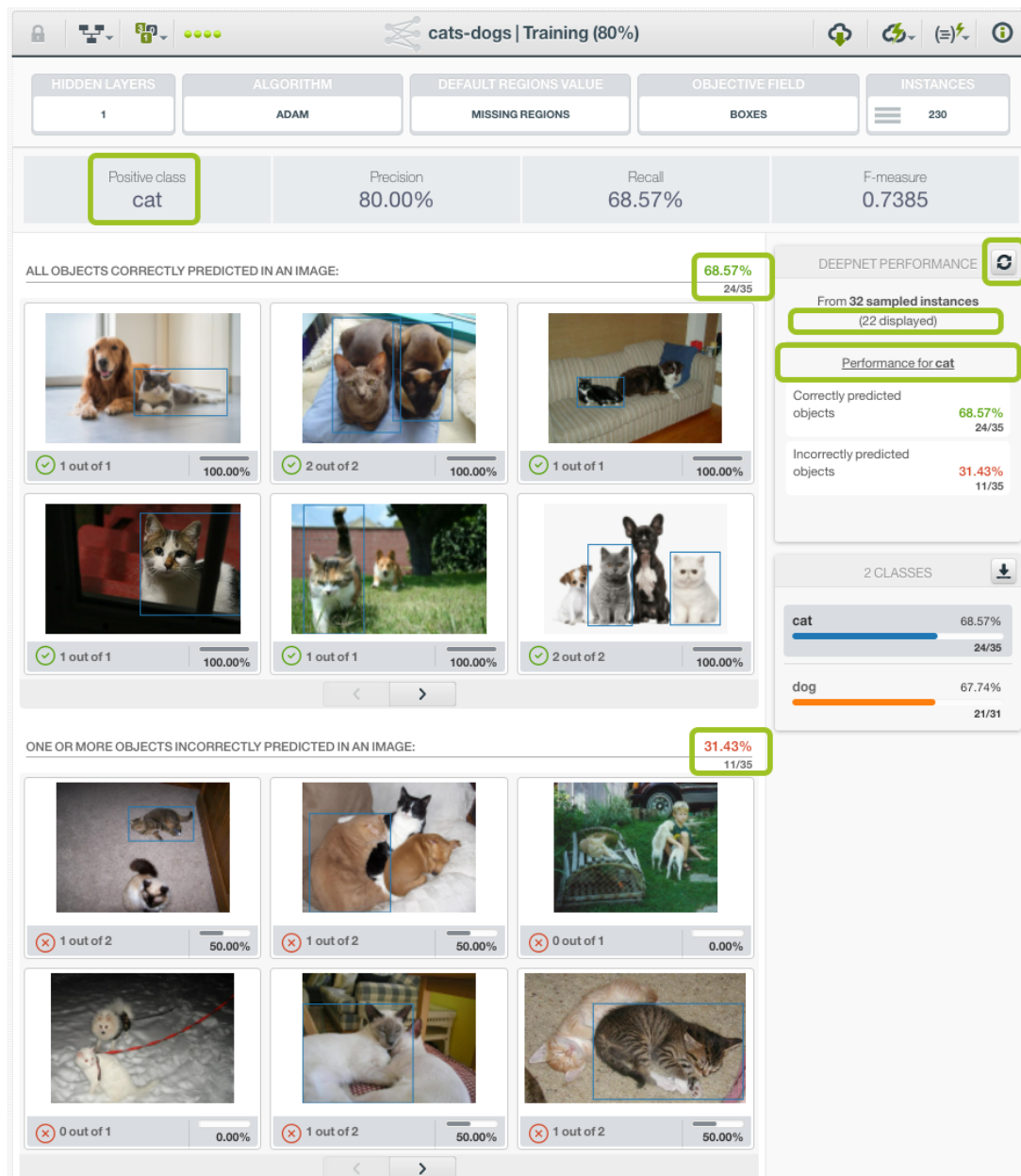


Figure 5.8: Object detection visualization showing one class

To deselect a class, or to reset the Image Results to include all classes, use the reset icon in the Performance Panel heading.

### 5.3 Inspecting Predictions

Inspecting the predictions in the holdout set can help understand an object detection deepnet. With different combinations of the viewing controls in the closeup view (Subsection 5.2.2.1, users can carefully compare the predictions and examine the overlaps of the predicted regions and ground truth regions.

In Figure 5.9, the object detection deepnet has 230 instances and its objective field is BOXES. Its precision is 78.95%, recall 68.18% and F-measure 0.7317. In its Performance Panel, the holdout set has 32 instances (images). It predicted 45 objects correctly out of 66. There are two classes, with the count of each class showing in the Class List.

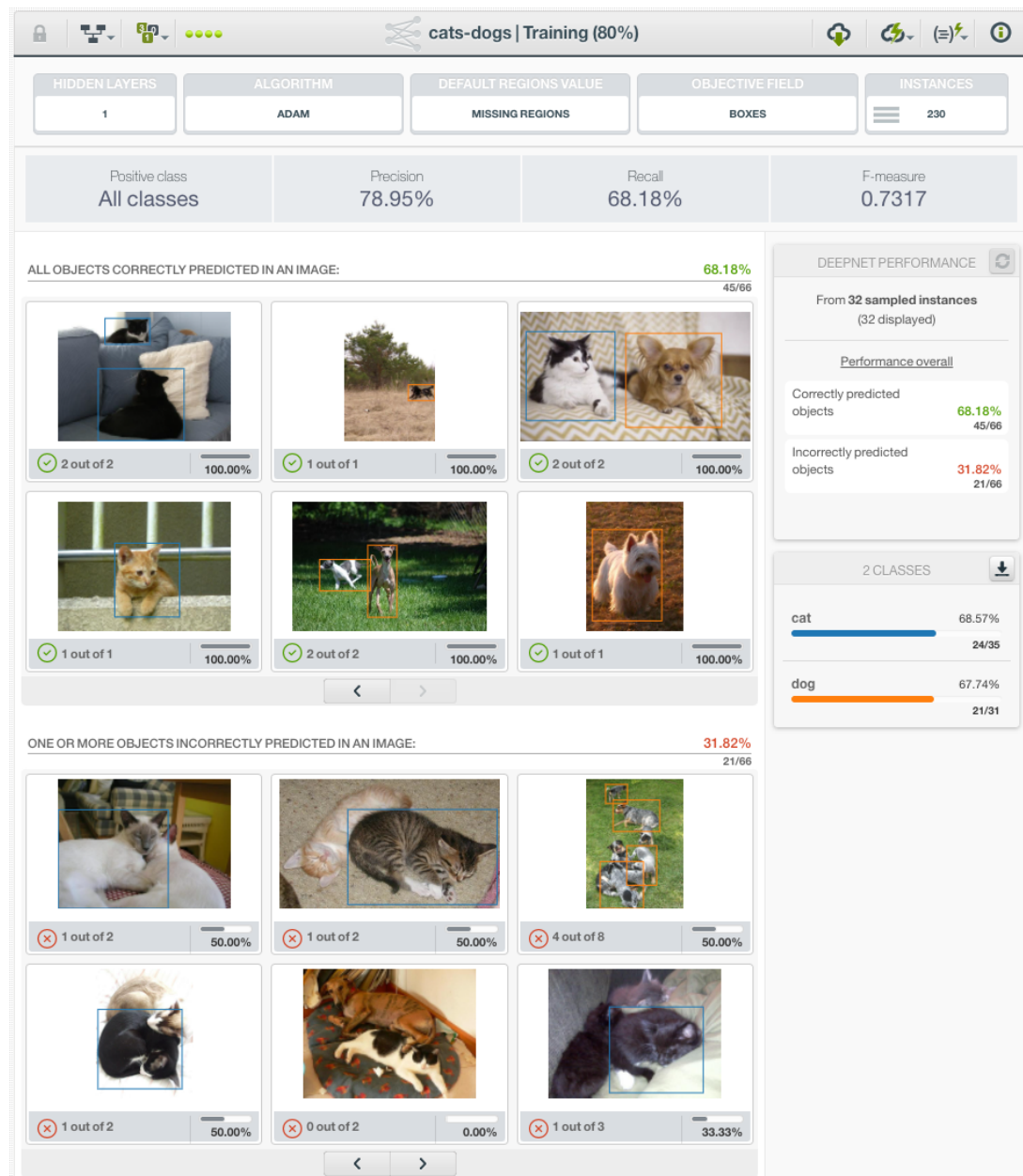


Figure 5.9: Object detection deepnet visualization

Clicking on any image will bring up the closeup view of the image. By default, the closeup view shows the correctly predicted regions and their prediction scores, as indicated by the highlighted checkbox on the right in Figure 5.10.

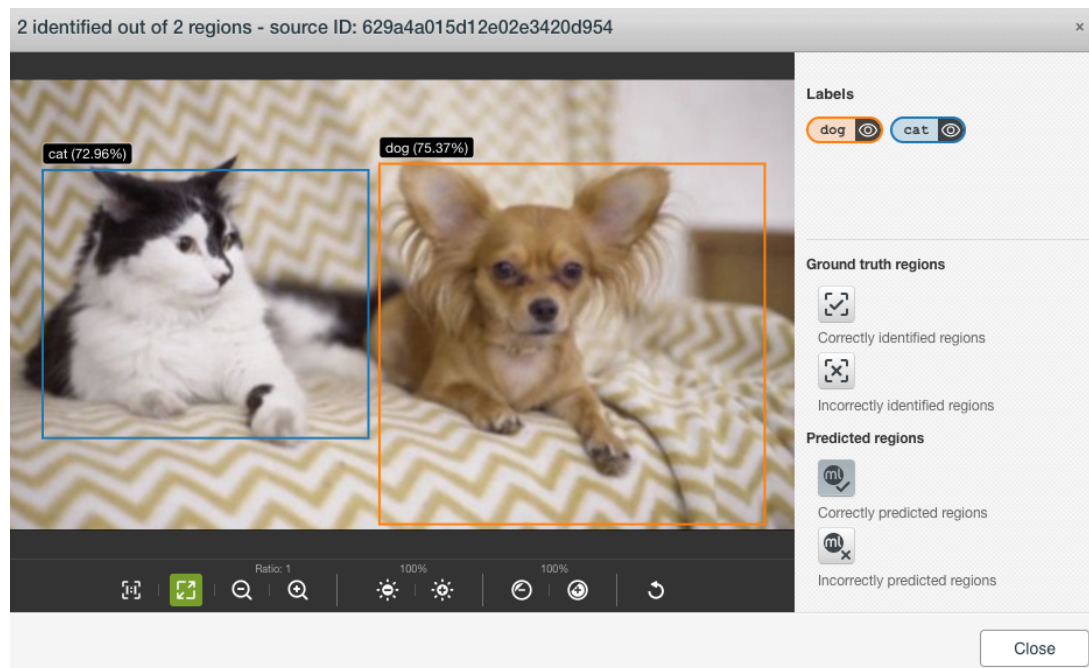


Figure 5.10: Correctly predicted regions

By also selecting the **Correctly identified regions** checkbox, it shows the ground truth regions. We then can compare the prediction regions and ground truth regions and see how they overlap each other in Figure 5.11.

In this particular image shown in fig:od-viz-closeup1-compare, both objects were predicted correctly, so there are no incorrectly identified regions, nor incorrectly predicted regions. We can also hide a class of objects using the class buttons on the top right.

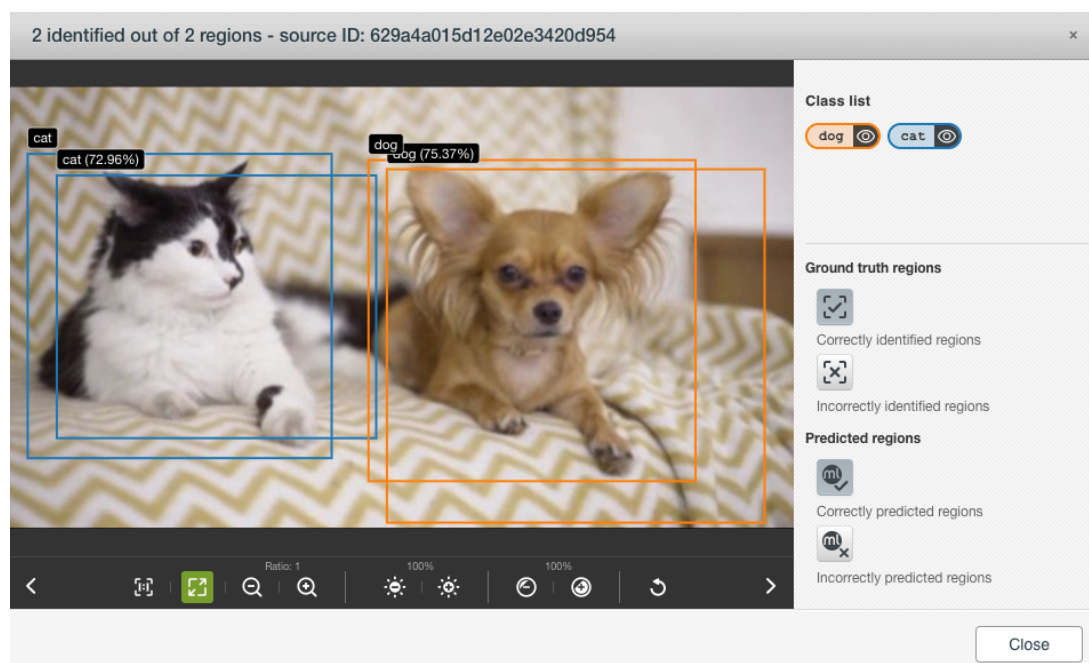


Figure 5.11: Comparing correctly predicted regions with their ground truth

Now click on an image in the lower subsection of the Image Results, as shown in Figure 5.12. Note the prediction scores on the predicted regions.



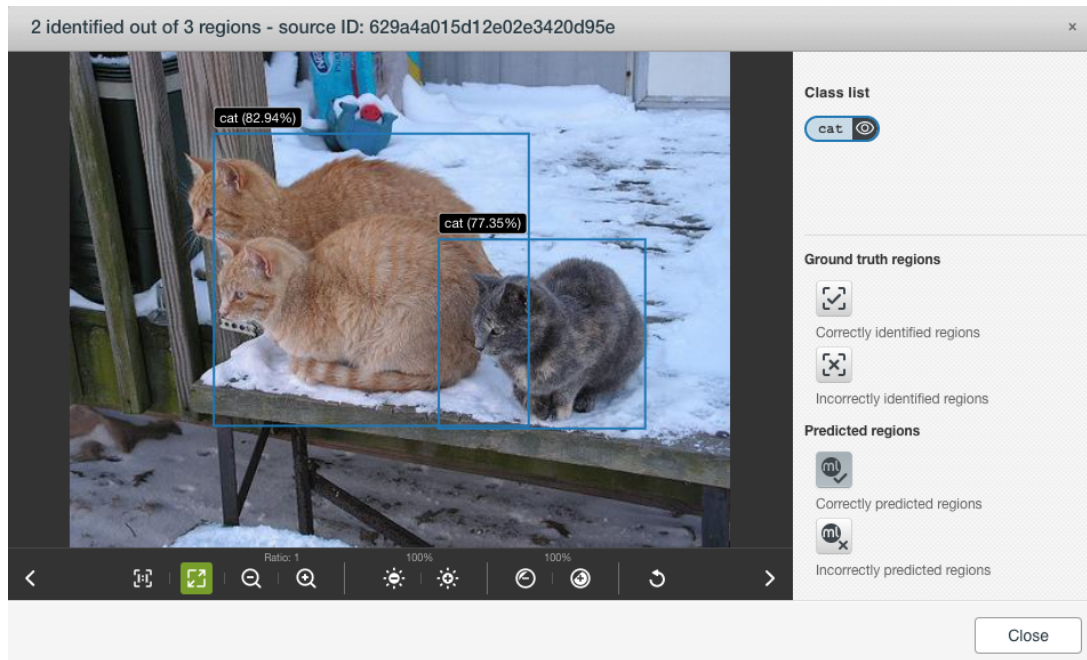


Figure 5.12: Showing the correctly predicted regions

There are two correctly predicted regions in this image, but there are actually three cats. We can compare the predicted and ground truth regions by selecting also the **Correctly identified regions** checkbox, shown in Figure 5.13

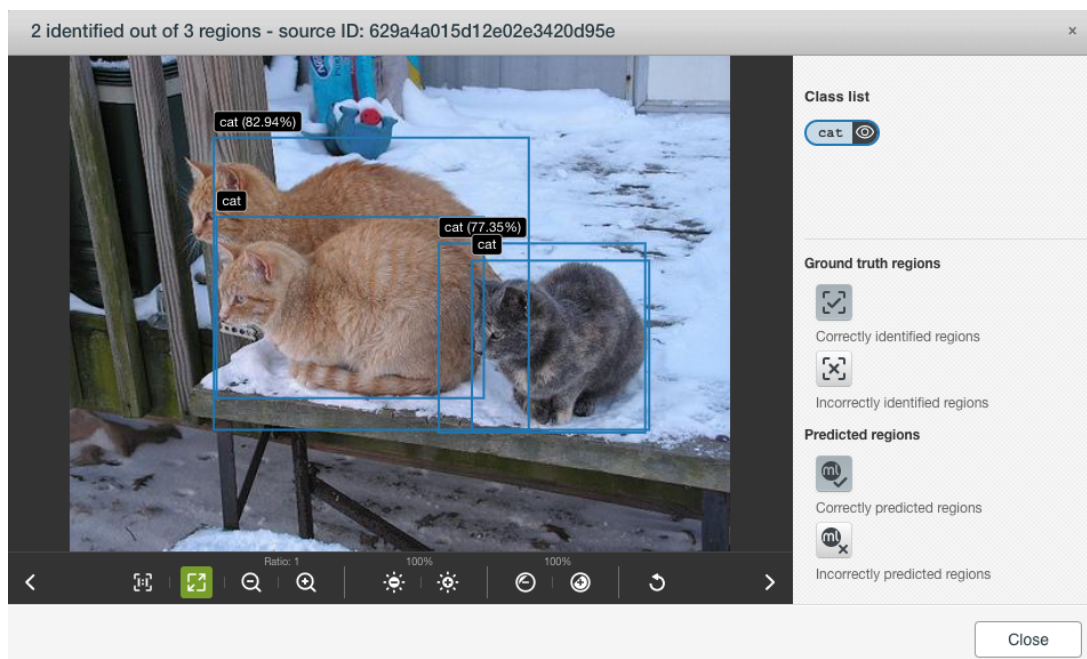


Figure 5.13: Comparing correctly predicted regions with its ground truth

Or we can just view all ground truth regions by deselecting the **Correctly predicted regions** checkbox and selecting both **Correctly identified regions** and **Incorrectly identified regions** checkboxes, as shown in Figure 5.14.

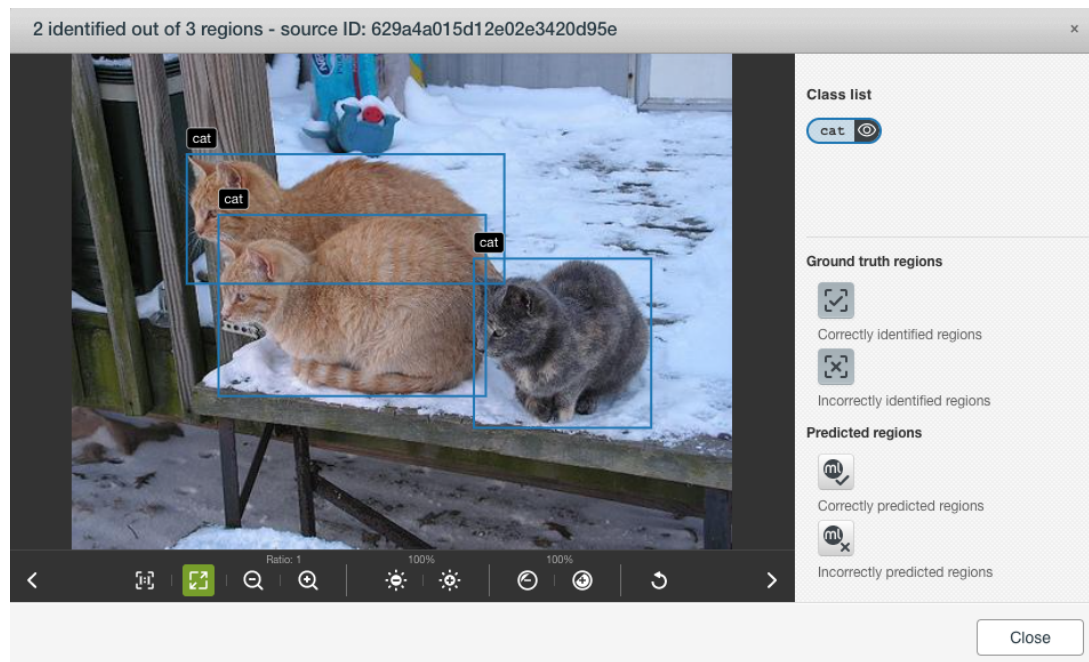


Figure 5.14: Showing all ground truth regions

With these checkbox combinations, it's clear that the object detection deepnet missed the cat in the back.

## Object Detection Evaluations

BigML provides object detection evaluations for users to measure the performance of object detection models. Evaluations not only produce estimates of a model's performance, but also provide a framework to compare models to help identify the ones with the best performance on different data.

### 6.1 Introduction

The basic idea behind evaluations is to take some test data different from the one used to train the model and create a prediction for every instance. Then compare the actual **objective field** values of the instances in the test data against the predictions and compute several performance measures based on the correct results as well as the errors made by the model. One thing is important: the test data has to be different from the training data, so that the model makes predictions for instances it has never seen before.

A simple way to obtain test data is to *split* a **dataset** into two subsets: a training set and a test set. You can easily do this from BigML Dashboard by using the 1-CLICK RANDOM TRAINING|TEST from the dataset view that randomly selects *80% of the instances for training* and the rest *20% for testing*, as shown in **Figure 6.1**.

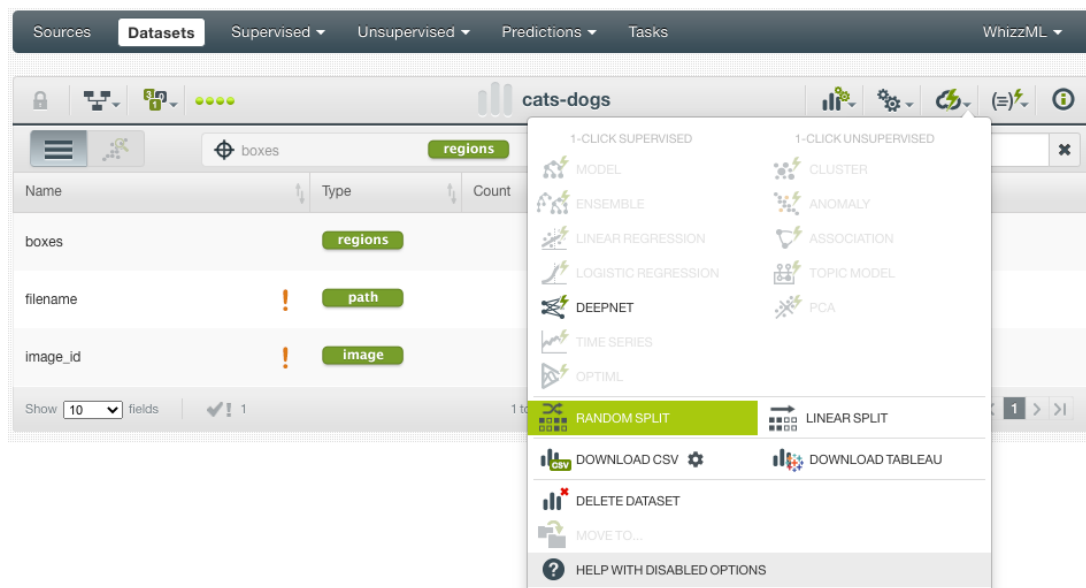


Figure 6.1: 1-click random split of a dataset

Alternatively, you can configure the splitting percentages as shown in **Figure 6.2**.



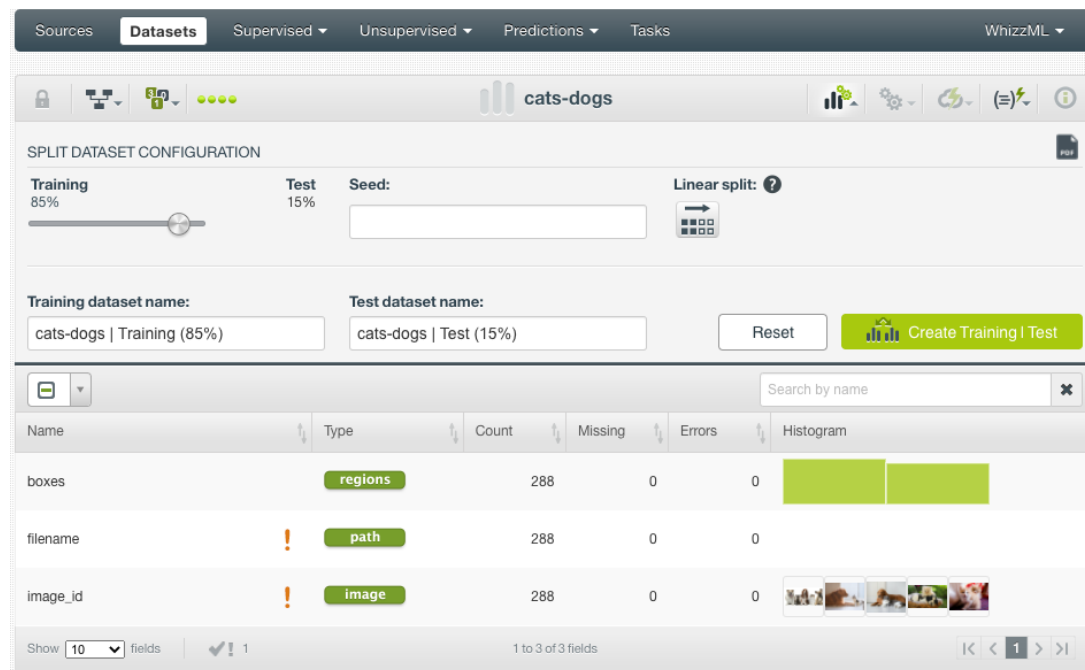


Figure 6.2: Configuring a random split of a dataset

The **evaluation list view** in the BigML Dashboard can be found in the SUPERVISED tab under the main menu (Figure 6.3.) This view contains all evaluations sorted by creation date so most recent evaluations are found at the top of the list. Sort the evaluations by **Name**, by **Type**, by **Performance**, by **Age** (time since the evaluation was created), by **Dataset Size** (the test dataset size) or by **Instances** (number of instances in the test dataset). The icons on the left of the evaluations names in the **evaluation list view** allow you to go to the original resources used to create the evaluation (e.g. object detection deepnets or datasets). In the evaluation list view, you can search evaluations by name by clicking the SEARCH menu option in the top right corner.

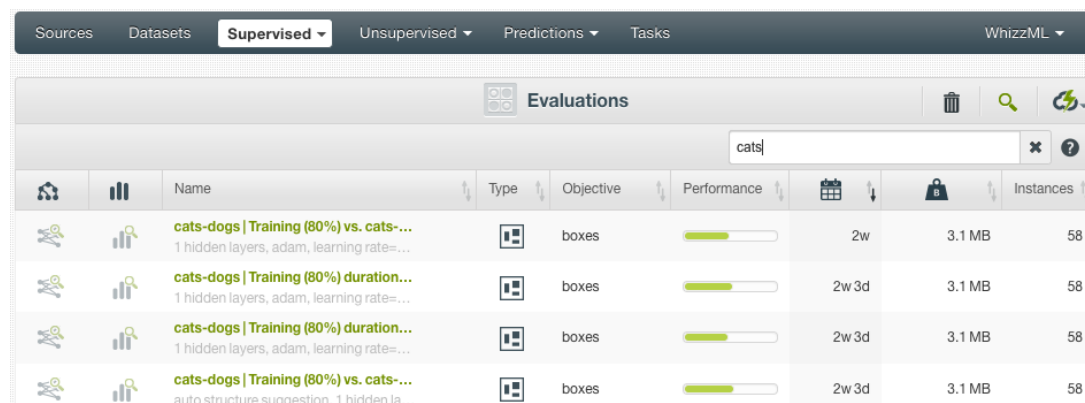


Figure 6.3: Evaluation list view

Look for the icon shown in Figure 6.4, which is used to represent an object detection evaluation in the evaluation list view.



Figure 6.4: Object detection evaluation icon

All object detection evaluation metrics are explained in [Section 2.3](#).

## 6.2 Creating Object Detection Evaluations

Two resources are required to create an object detection evaluation: an object detection deepnet and a test dataset. Use the **EVALUATE** option in the **1-click action menu** from the object detection visualization page, as shown in [Figure 6.5](#)).

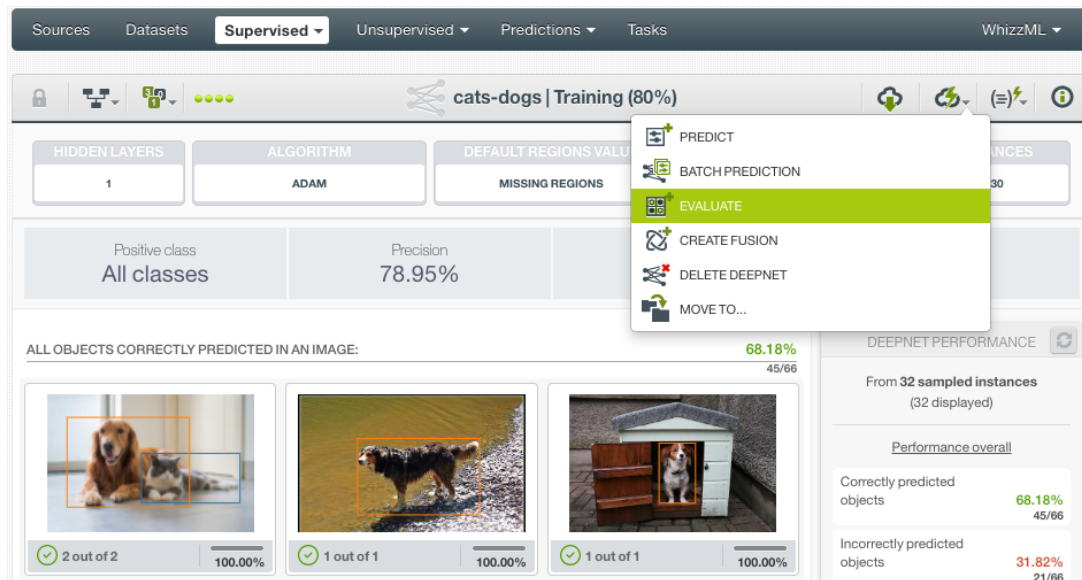


Figure 6.5: Creating an evaluation from an object detection deepnet

Or use the **EVALUATE** option in the **pop up menu** from the deepnet list view ( [Figure 6.6](#)).

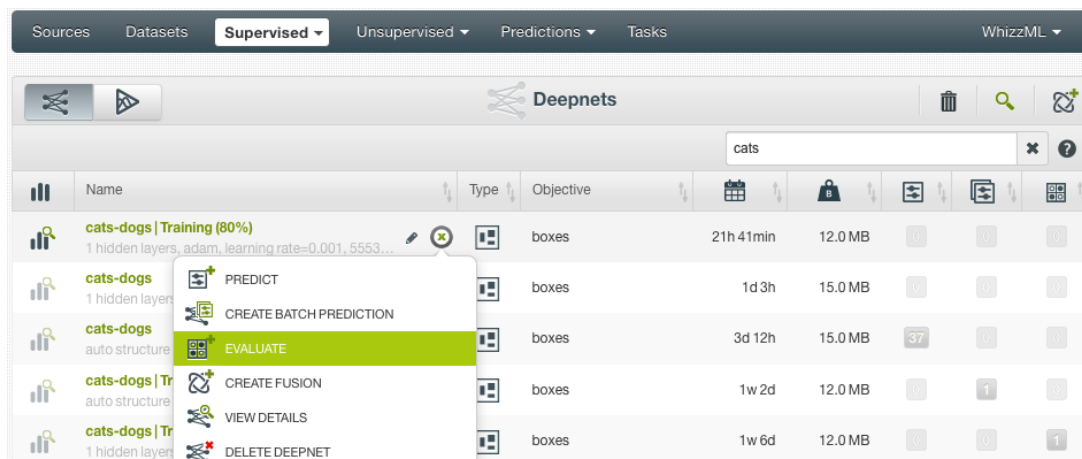


Figure 6.6: Creating an evaluation from the deepnet list view

Using either of the options will reach the **New Evaluation** view where the object detection deepnet will be pre-filled in the model selector and you only have to choose the test dataset. If you previously split your original dataset into two subsets, one for training and another for testing (see [Section 6.1](#)), BigML will automatically select the corresponding test dataset. Then click the **Evaluate** green button to create the evaluation, shown in [Figure 6.7](#).

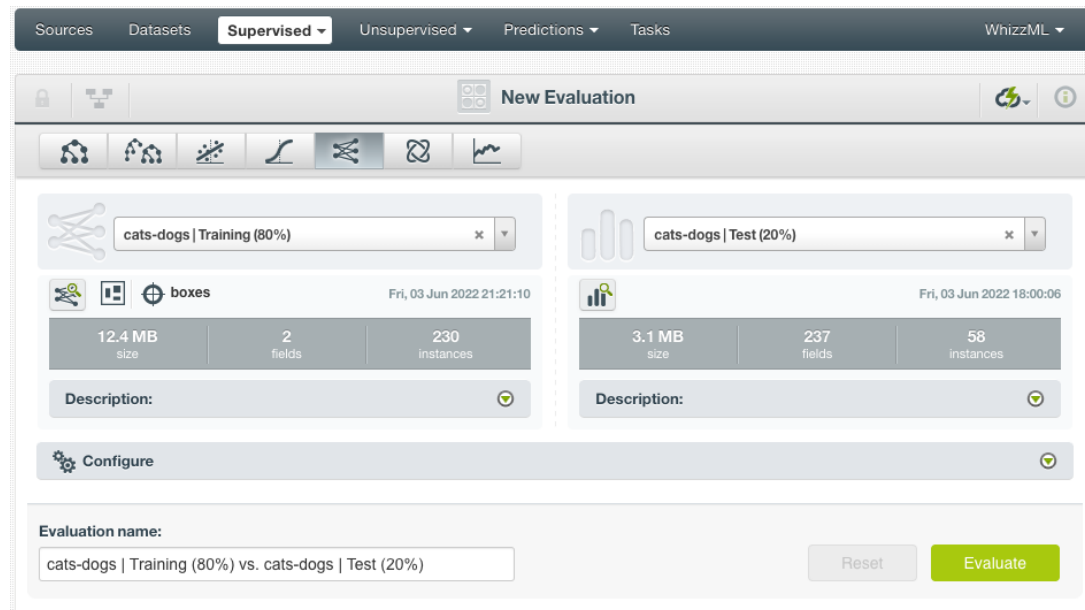


Figure 6.7: Creating a new evaluation

You can configure some options when creating an object detection evaluation, such as excluding fields, fields mapping and sampling. They are common among all evaluations and are explained in section Evaluation Configuration Options of the [Classification and Regression with the BigML Dashboard](https://static.bigml.com/pdf/BigML_Classification_and_Regression.pdf)<sup>1</sup>[2].

## 6.3 Visualizing Object Detection Evaluations

An object detection evaluation page is composed of two parts: the **Main control** to select a view and an IOU threshold, and the **View area** which presents evaluation metrics in different formats.

<sup>1</sup>[https://static.bigml.com/pdf/BigML\\_Classification\\_and\\_Regression.pdf](https://static.bigml.com/pdf/BigML_Classification_and_Regression.pdf)

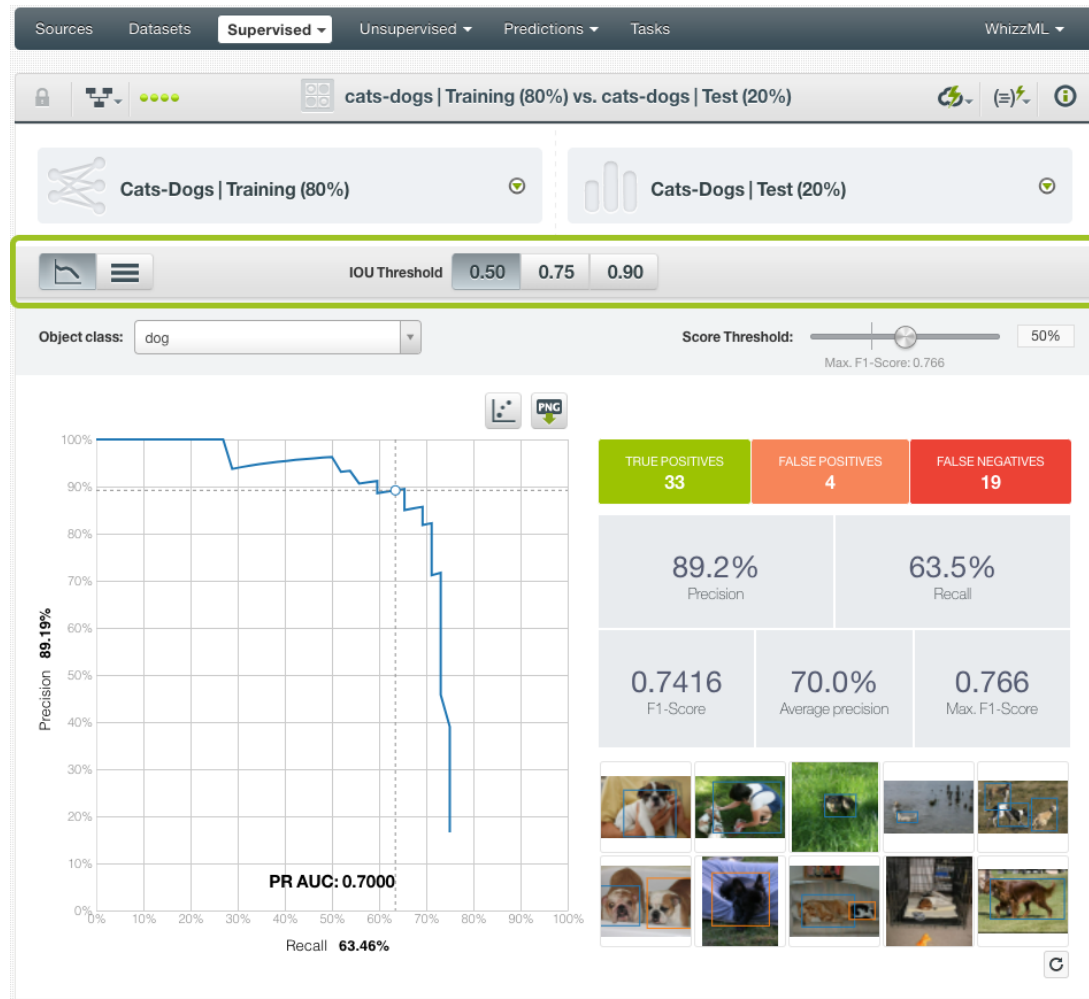


Figure 6.8: Evaluation main control

### 6.3.1 Main Control

The **Main control**, marked by the green rectangle in Figure 6.8, is below the model and dataset names and above the View area. It provides five tabs. On the left are two tabs for evaluation views. In the middle are three tabs for IOU thresholds (Subsection 2.3.1).

There are two evaluation views: **Class detail** and **Summary**. Select one of them to go to a view. Class detail is selected by default. BigML provides three IOU thresholds: 0.5, 0.75 and 0.9. They cover a wide range of prediction qualities. Press an IOU tab to select one. 0.5 is selected by default.

All metrics below the main control are calculated based on the selected IOU. Naturally, when the IOU is high, such as 0.9, there are fewer correct predictions, thus the performance under 0.9 IOU is lower than that under 0.5 IOU.

### 6.3.2 Evaluation View - Class Detail

The **Class detail** view is selected by default. Its view area has four panels, as shown in Figure 6.9.

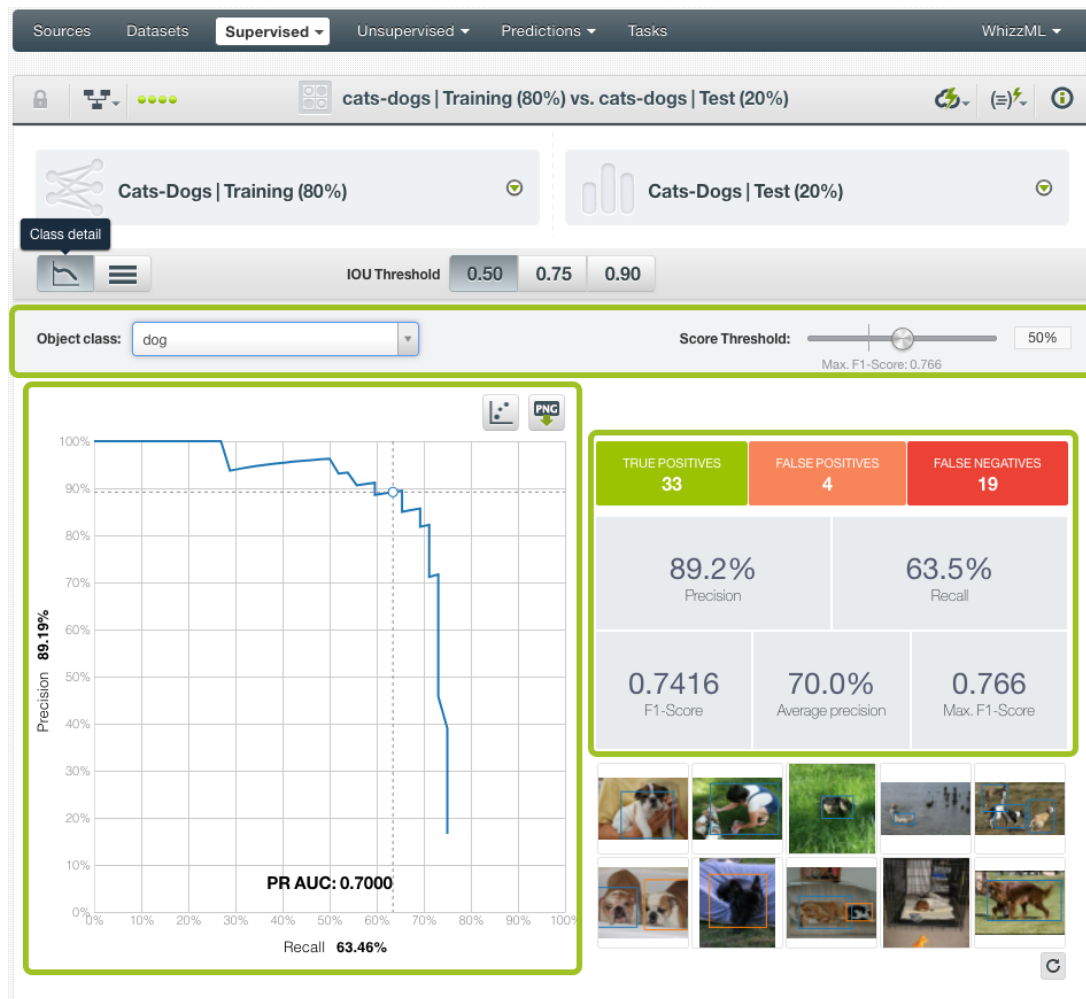


Figure 6.9: Class detail view

### 6.3.2.1 Selection

The **Selection** panel is at top of the view area and just below the Main control. It provides a drop-down list for selecting a class of the objects. On the right is a slider for selecting a prediction score threshold (Subsection 2.3.2). The default score is set to 50%.

The score threshold can be changed by dragging the knob on the slider to a value between 0% to 100%. Or a value can be entered in the input box at the slider's right end.

Lowering the score threshold will increase the true positives hence improving the recall, but the precision may suffer. And vice versa when increasing the threshold.

There is a vertical bar on the score threshold slider, It marks the threshold value that achieves the maximum F1-Score. The maximum F1-Score value is displayed under the vertical bar.

### 6.3.2.2 Chart

The **Chart** is on the left side of the view area. It shows the Precision-Recall Curve, also known as PR Curve.

The Precision-Recall Curve visually represents the trade-off between both measures for the selected class. Precision and recall are inversely related, i.e., for the same object detection deepnet you can increase recall by lowering the threshold for the selected class, but it will usually result in a decrease in precision, and vice versa. The formulas of both measures are in Subsection 2.3.4.

A high precision and a high recall are represented by points near the upper right corner of the chart, so the greater the area under the precision-recall curve the better. BigML provides two different area

calculations:

- **PR AUC:** the Area Under the Curve (AUC) is calculated taking into account the exact curve shape (Figure 6.10).

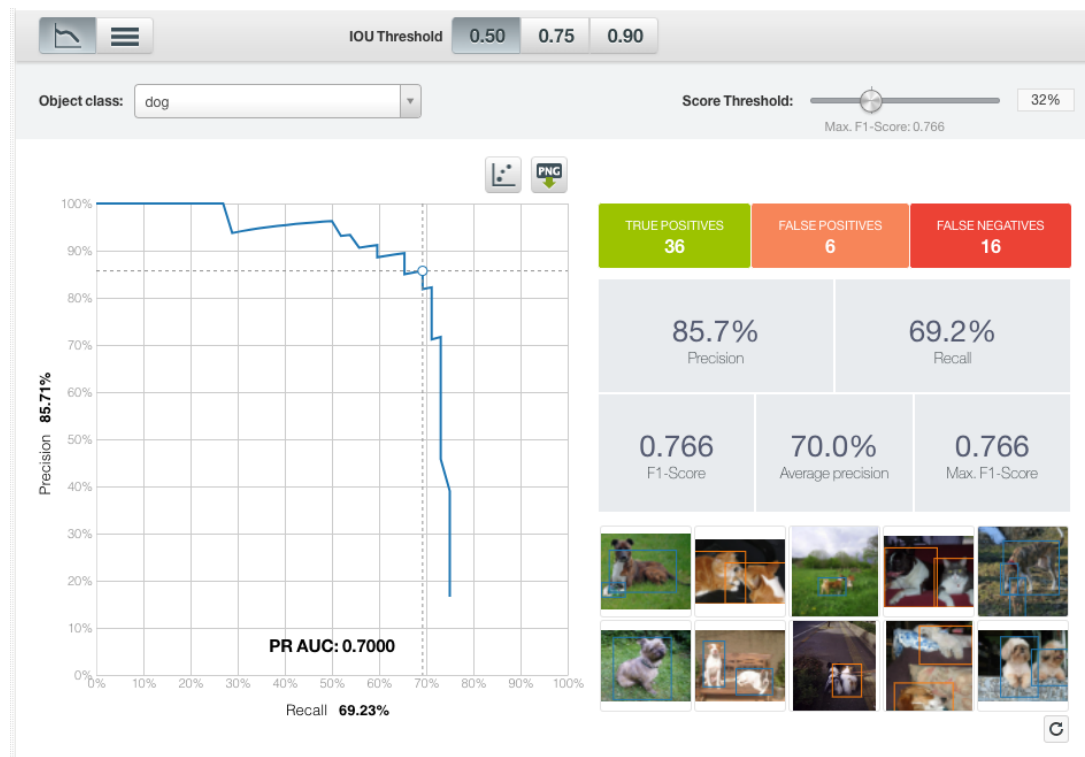


Figure 6.10: PR AUC: Area Under the Curve of a PR curve

- **PR AUCH:** the Area Under the Convex Hull is calculated taking into account the convex shape of the curve where no other points lay above the curve. You can visualize it by clicking on the AUCH icon at the top right of the chart, shown in Figure 6.11.

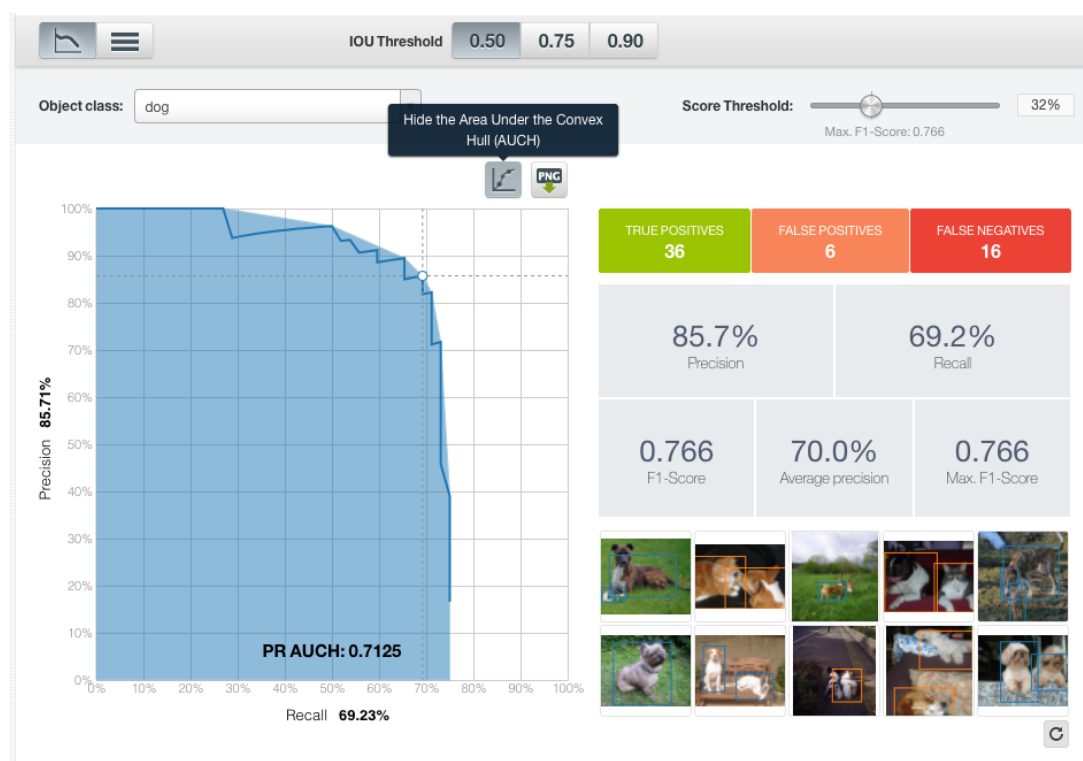


Figure 6.11: PR AUCH: Area Under the Convex Hull of a PR curve

At the top right of the Chart are two icons, one is the AUCH icon. The export icon is for exporting the chart as an image, either with legends or without.

### 6.3.2.3 Metrics

The **Metrics** panel is on the right of the Chart and below the Selection panel. It shows all the metrics for by the selected class. Many of the metrics are also based on the score threshold.

On the first row the panel are three counts with colored backgrounds: True Positives, False Positives and False Negatives (Subsection 2.3.3). Changing the score threshold will change their values.

On the second row are Precision and Recall (Subsection 2.3.4), which are also affected by the score threshold.

On the third row are the F1-Score (Subsection 2.3.5), the Average Precision (Subsection 2.3.6) and the Maximum F1-Score. The value of F1-Score changes when the score threshold changes. The maximum F1-Score is the maximum value of F1-Score among all score threshold values. The threshold value that achieves the maximum F1-Score is shown by the position of a vertical bar on the threshold slider.

### 6.3.2.4 Image Preview

The **Image preview** panel is at the lower right of the view area. It shows sampled images from the test dataset that was used for the evaluation. The panel has two rows and displays up to 10 images. Clicking on the refresh icon on the bottom right will retrieve different images from the test dataset.

Clicking on an image in the Image Preview Panel will bring out a closeup view of the image, allowing for prediction inspection.

As seen in Figure 6.12, there are viewing controls in the closeup view, based on classes, ground truth and prediction. Users can use different combinations of the viewing controls to show or hide certain regions or certain classes. This allows easy and clear comparison between prediction results and ground truth.



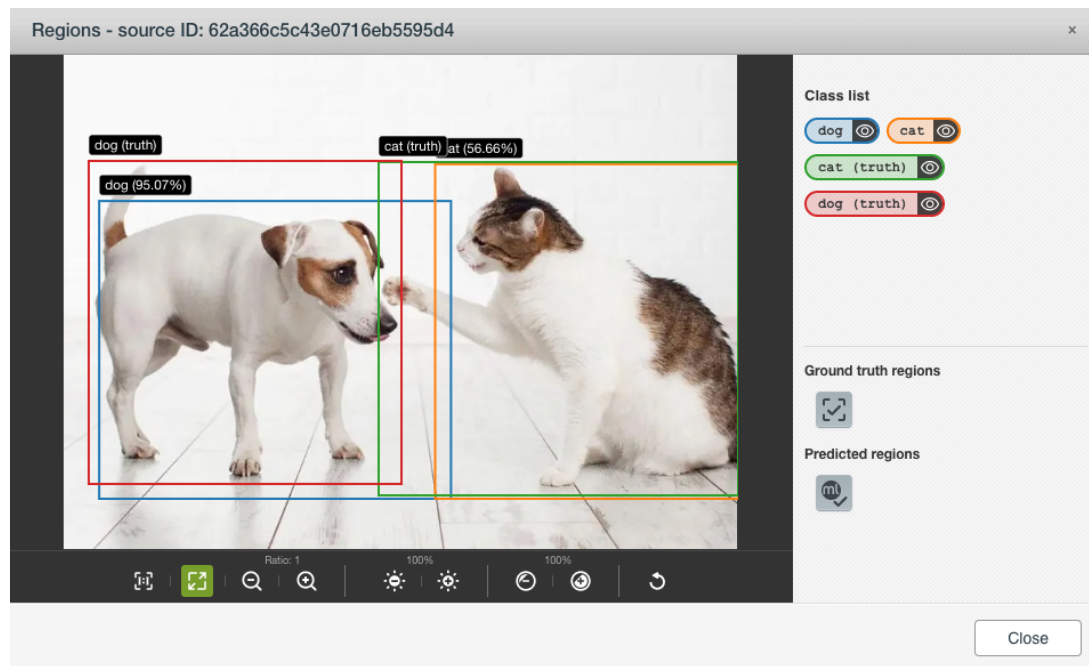


Figure 6.12: Closeup view of an image in Image Preview Panel

### 6.3.3 Evaluation View - Summary

Clicking on the **Summary** tab in the main control will get to the **Summary** view, shown in Figure 6.13.

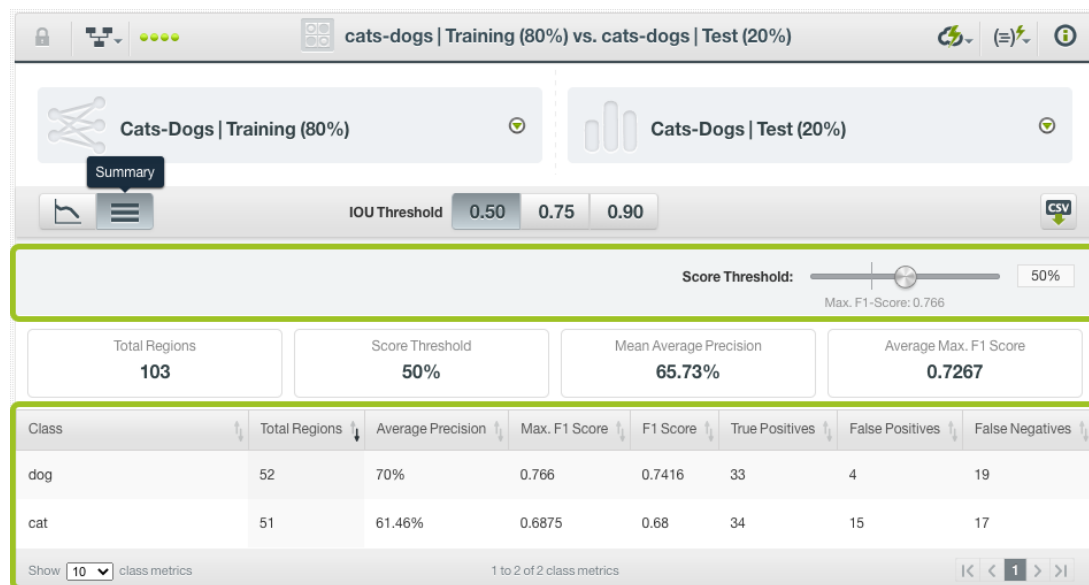


Figure 6.13: Summary view

There are three panels in the view area of a **Summary** view.

#### 6.3.3.1 Selection

The **Selection** panel is at the top of the View area and just below the Main control. It provides a slider for selecting a prediction score threshold (Subsection 2.3.2). The default score is set to 50%, or to whatever was changed in the other view.

The threshold slider is the same one as in the Class detail view (Subsection 6.3.2.1). Changing the threshold value in one view will also change the value in another view.

### 6.3.3.2 Stats

The **Stats** panel is below the Selection panel. It shows the total number of regions, the current prediction score threshold, the mean average precision and the average maximum F1–Score.

The total number of regions is the total number of objects annotated in the test dataset, including all classes. The mean average precision (mAP, [Subsection 2.3.6](#)) is the sum of all APs (average precisions) divided by the number of classes. The average maximum F1–Score ([Subsection 2.3.5](#)) is the sum of all maximum F1–Scores divided by the number of classes.

### 6.3.3.3 Table

The **Table** panel shows a table of evaluation metrics by all classes. The metrics include Total Regions, Average Precision, Maximum F1–Score, F1–Score, True Positives, False Positives and False Negatives. Each row shows the metrics by one class, each column shows one metric. Every column is sortable. By default, the table is sorted by the **Total Regions**. There are pagination controls at the bottom of the table, which are useful when the number of classes is more than 10.

When the score threshold is changed, F1–Score, True Positives, False Positives and False Negatives change as well.

# Object Detection Predictions

## 7.1 Introduction

The ultimate goal in building an object detection deepnet is being able to make predictions with it, or to detect objects with it. In BigML, you can make predictions for *single instances* or for *many instances in batch*. Each prediction comes with a measure called *prediction score* or simply *score* (Subsection 2.3.2), which indicates the prediction confidence.

Object detection predictions are saved under the CLASSIFICATION, REGRESSION & DETECTION option of the PREDICTIONS tab in the main menu (Figure 7.1).

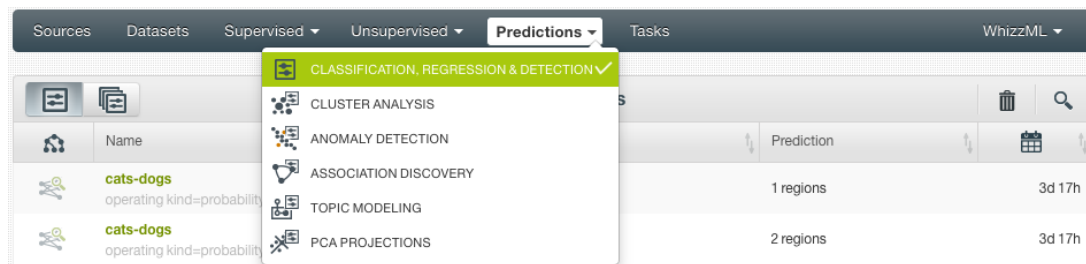
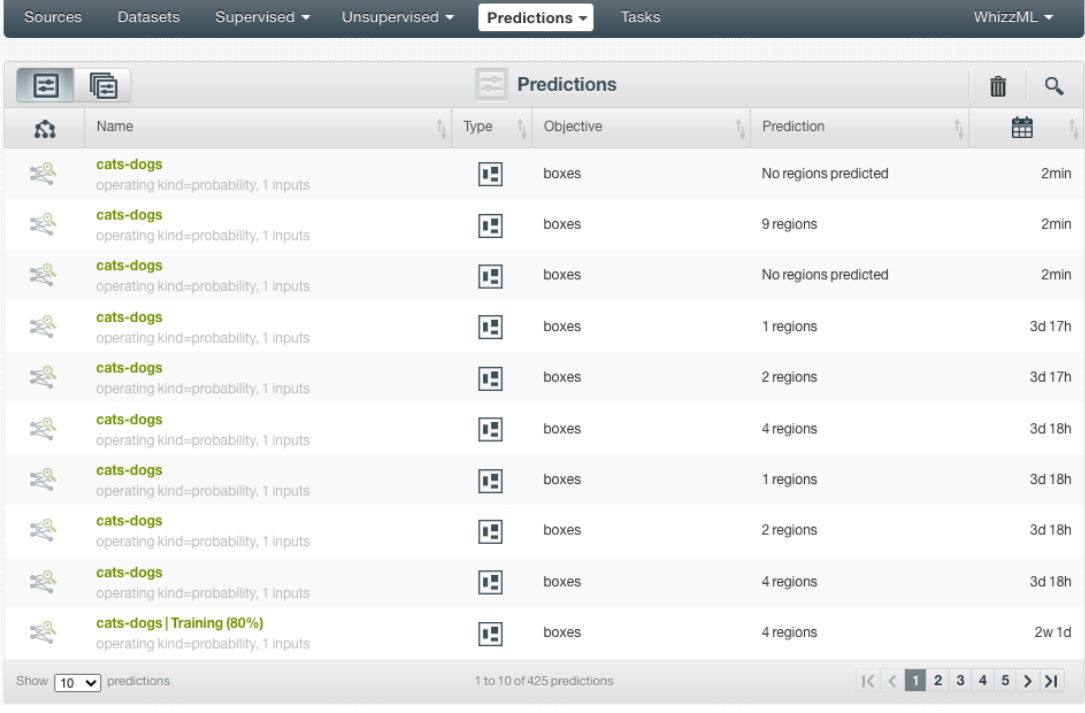


Figure 7.1: Menu options under Predictions

Selecting the menu option will lead to the prediction list view, as shown in Figure 7.2. You can **search** predictions by name by clicking on the search option on the top menu. In the predictions list view, each prediction shows the **object detection** icon used for the prediction, the **Name** of the prediction, the **Objective** (objective field name), the **Prediction** (the prediction result), and the **Age** (time since the prediction was created).



Name	Type	Objective	Prediction	
cats-dogs operating kind=probability, 1 inputs	boxes	boxes	No regions predicted	2min
cats-dogs operating kind=probability, 1 inputs	boxes	boxes	9 regions	2min
cats-dogs operating kind=probability, 1 inputs	boxes	boxes	No regions predicted	2min
cats-dogs operating kind=probability, 1 inputs	boxes	boxes	1 regions	3d 17h
cats-dogs operating kind=probability, 1 inputs	boxes	boxes	2 regions	3d 17h
cats-dogs operating kind=probability, 1 inputs	boxes	boxes	4 regions	3d 18h
cats-dogs operating kind=probability, 1 inputs	boxes	boxes	1 regions	3d 18h
cats-dogs operating kind=probability, 1 inputs	boxes	boxes	2 regions	3d 18h
cats-dogs operating kind=probability, 1 inputs	boxes	boxes	4 regions	3d 18h
cats-dogs   Training (80%) operating kind=probability, 1 inputs	boxes	boxes	4 regions	2w 1d

Figure 7.2: Prediction list view

Look for the icon of [Figure 7.3](#) in the **Type** column, which is used to represent an object detection prediction in the prediction list view.



Figure 7.3: Object detection prediction icon

On the top left of the prediction list view, there are two icons for selecting different kinds of predictions. Select the list for single predictions or the list of batch predictions by clicking on the corresponding icons, in [Figure 7.3](#) and [Figure 7.5](#).



Figure 7.4: Single prediction icon



Figure 7.5: Batch prediction icon

## 7.2 Creating Object Detection Predictions

BigML provides two options of prediction with object detection:

- PREDICT: to predict one single instance
- BATCH PREDICTION to predict multiple instances in batch

### 7.2.1 Single Predictions

Single predictions are the predictions performed by the object detection deepnet on a single instance, which is an image.

Follow these steps to create a single prediction:

1. While in the object detection visualization page, click PREDICT in the **1-click action menu**, as shown in Figure 7.6.

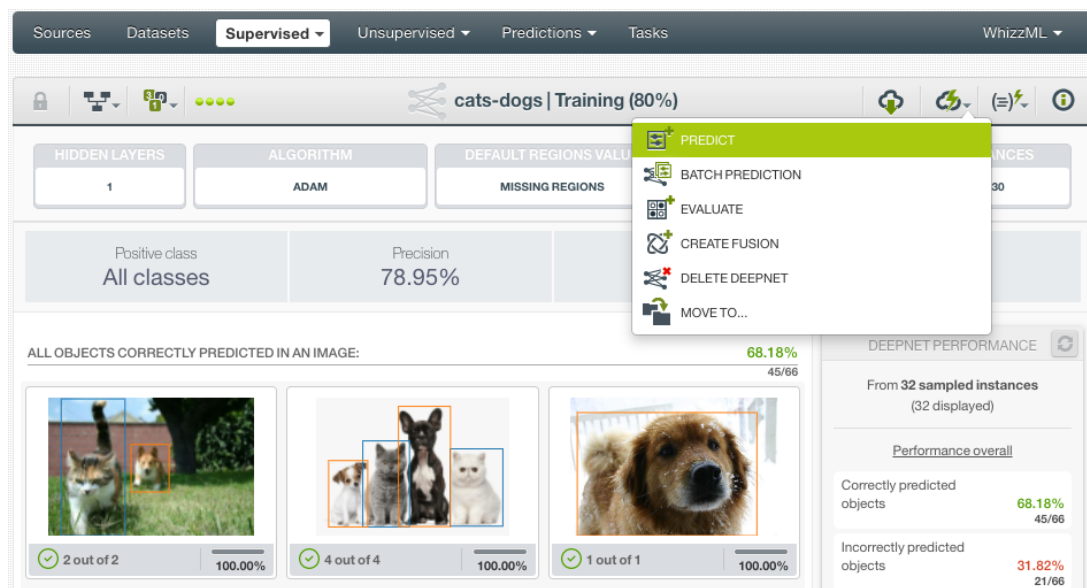


Figure 7.6: Predict using the 1-click action menu

Alternatively, click PREDICT in the **pop-up menu** in the deepnet list view. See Figure 7.7.

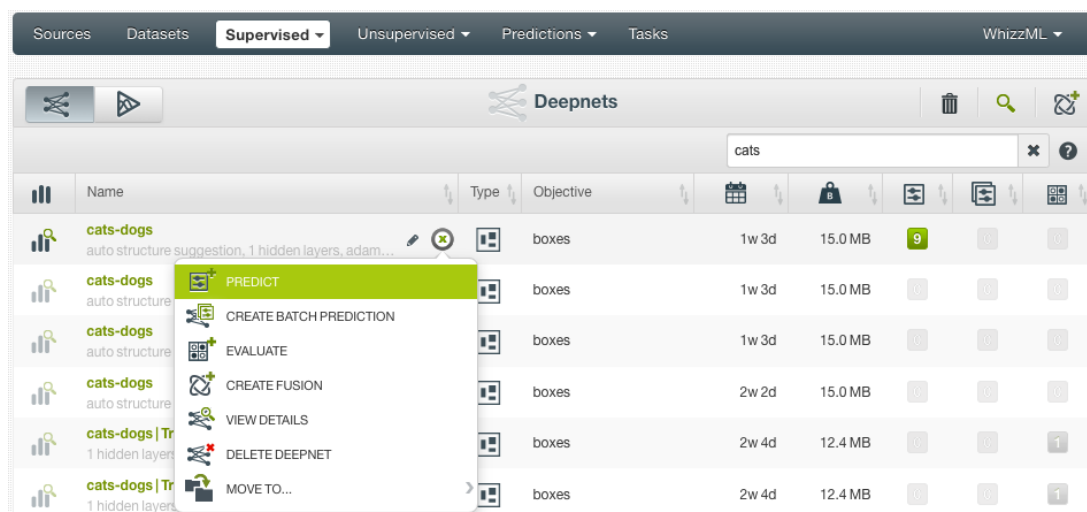


Figure 7.7: Predict using the pop-up menu

2. Either option will lead to the **prediction form** where you will find the input field used by object detection deepnet, which is an image field. As seen in [Figure 7.8](#), click on the input field box to select an image. Because this is a single prediction, an image is input by using a single image source. Clicking on the input box, single image sources available will be in the drop-down list. You can also use the search box to locate specific ones.

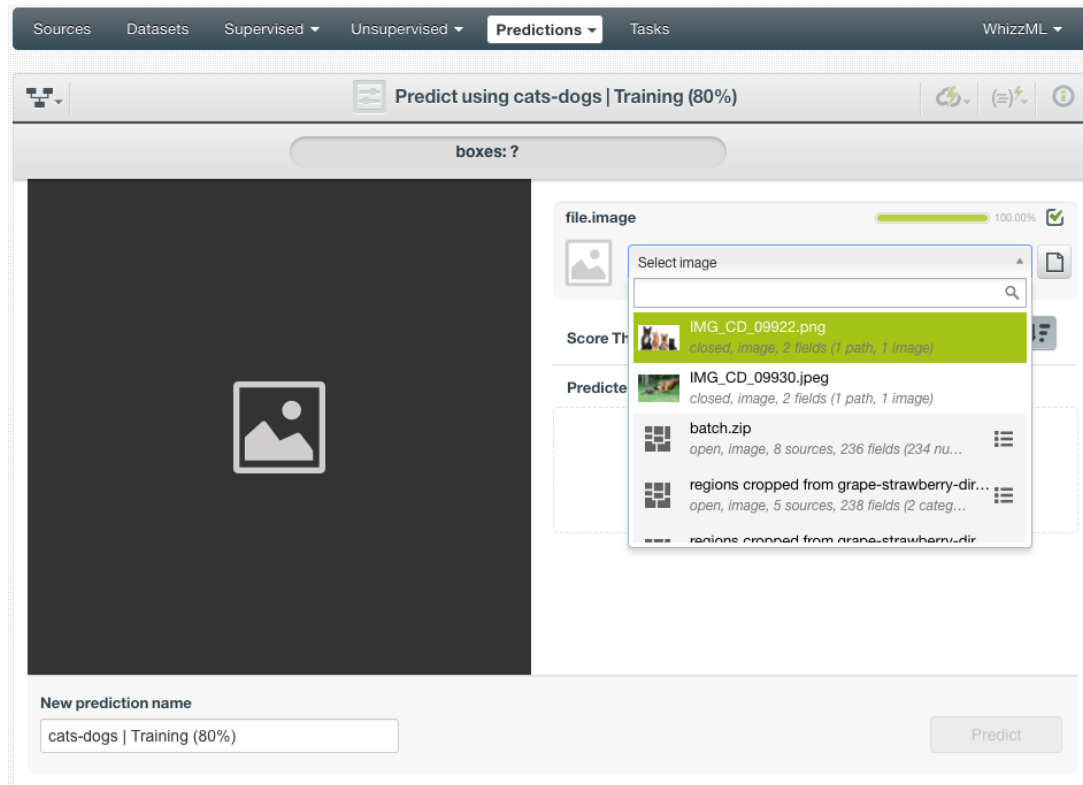


Figure 7.8: Select a single image source

3. Oftentimes single image sources were used for creating a composite source, and they became component sources of the composite source. Or an image was uploaded as a part of an archive file (zip/tar) which created a composite source. In those cases, the composite source will be shown in the drop-down list, along with an icon **List components**. In the example in [Figure 7.9](#), *batch.zip* is a composite source, click on the icon to show its component sources.

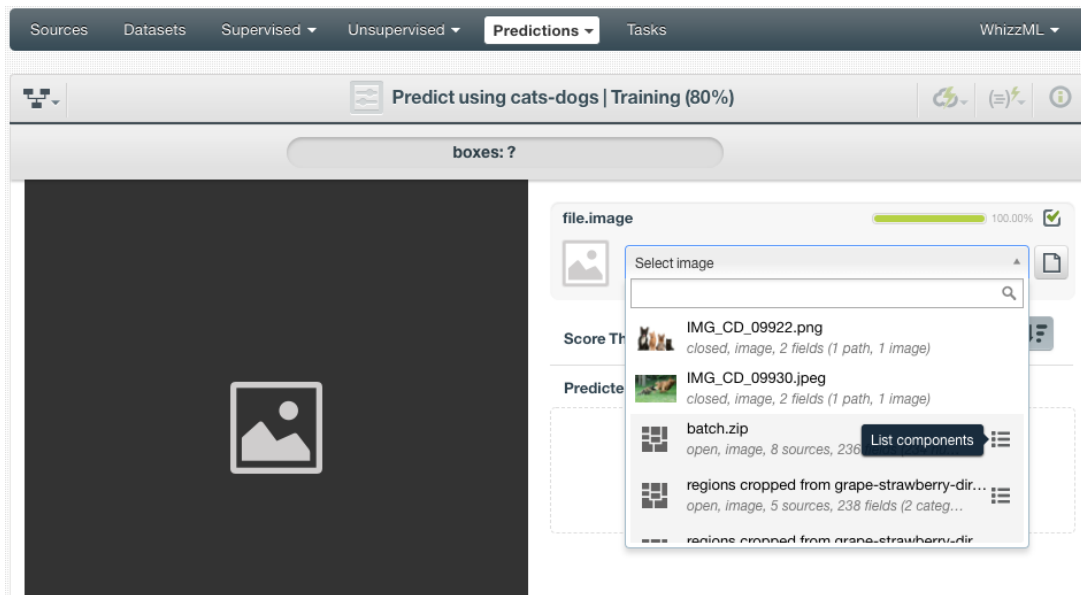


Figure 7.9: List the components of a composite source

After the component sources of the composite are listed, scroll the drop-down list to find the desired one, then click to select it, as shown in Figure 7.10.

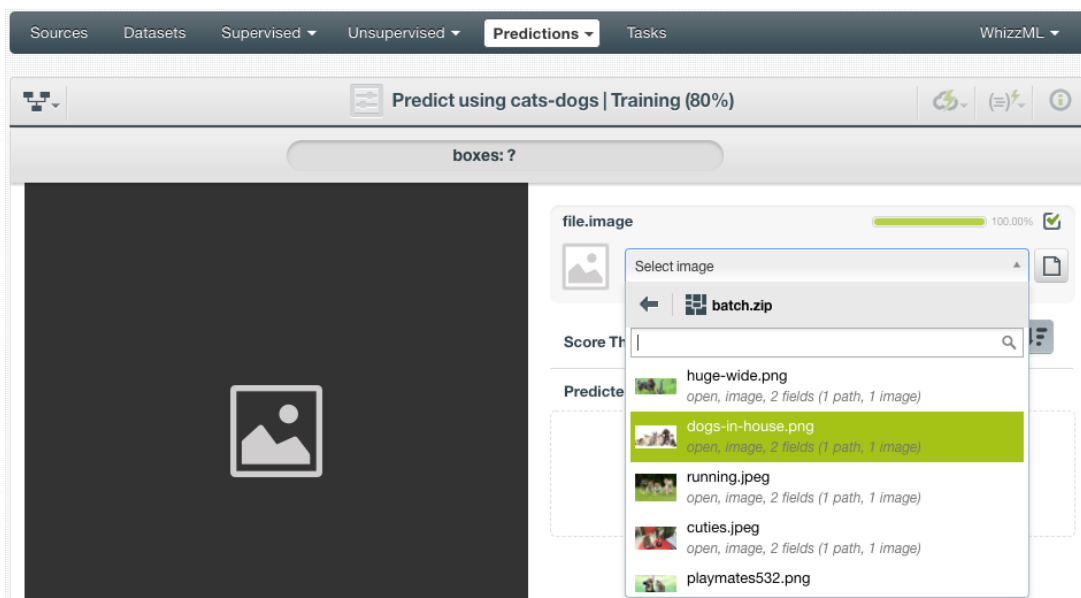


Figure 7.10: Select a component of a composite source

4. Another way to select an image is to load it from your local storage. Click on the **file** icon on the right of the image input box, it will bring a file browser and allow you to pick an image file in your local computer. This basically does an instant upload and creates a single image source.
5. Once an image is selected, it will be shown in the image view panel on the left, shown in Figure 7.11.



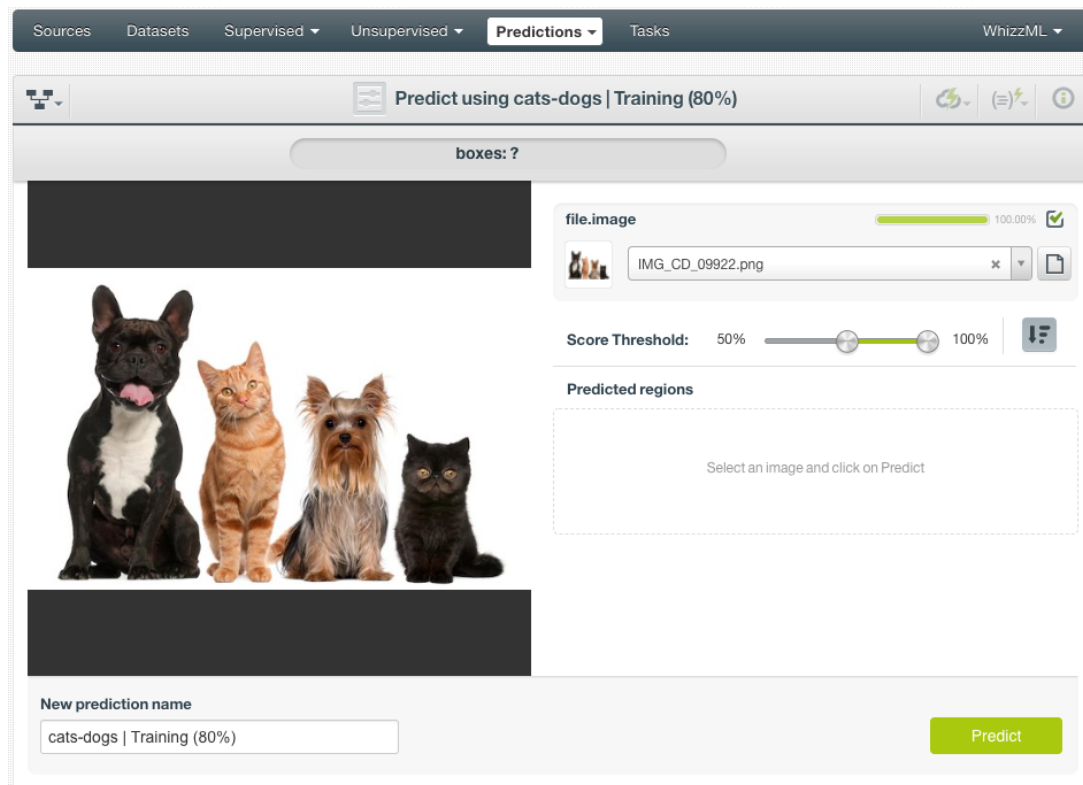


Figure 7.11: Image selected for a single prediction

Now you can select a prediction score threshold use the slider below the input box. The *prediction score* (Subsection 2.3.2), or simply *score*, measures how confident the object detection deepnet is about a prediction. The score threshold is a range and is used to filter out predictions. For instance, if the threshold is set to 30% to 65%, the result will only show predictions having a score within that range.

The default score threshold is set to 50% to 100%. Either end of the slider can be changed by dragging the respective knob. The threshold values shown are also input boxes, which can be used for entering threshold values.

6. After an image is selected and the score threshold set, click on the green button **Predict** to create a prediction for the image.

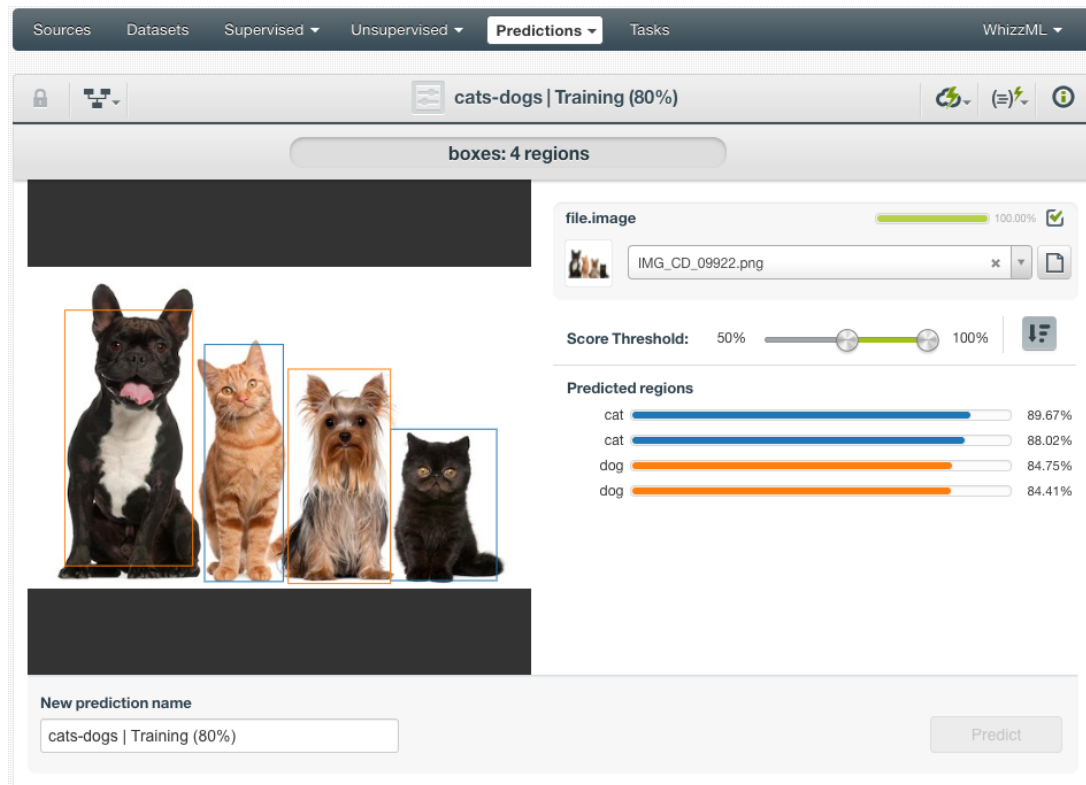


Figure 7.12: A single prediction

As shown in Figure 7.12, the predicted regions are listed under the threshold slider, along with their prediction scores. They are sorted by their scores by default. The sorting icon is at the right of the threshold slider. It can be clicked for disabling or enabling the sorting. If the sorting is off, the prediction regions will be grouped by classes.

7. Clicking on an image will bring a closeup view of the image along with the prediction results, as shown in Figure 7.13. We can click on the class labels on the right to hide or show certain classes.

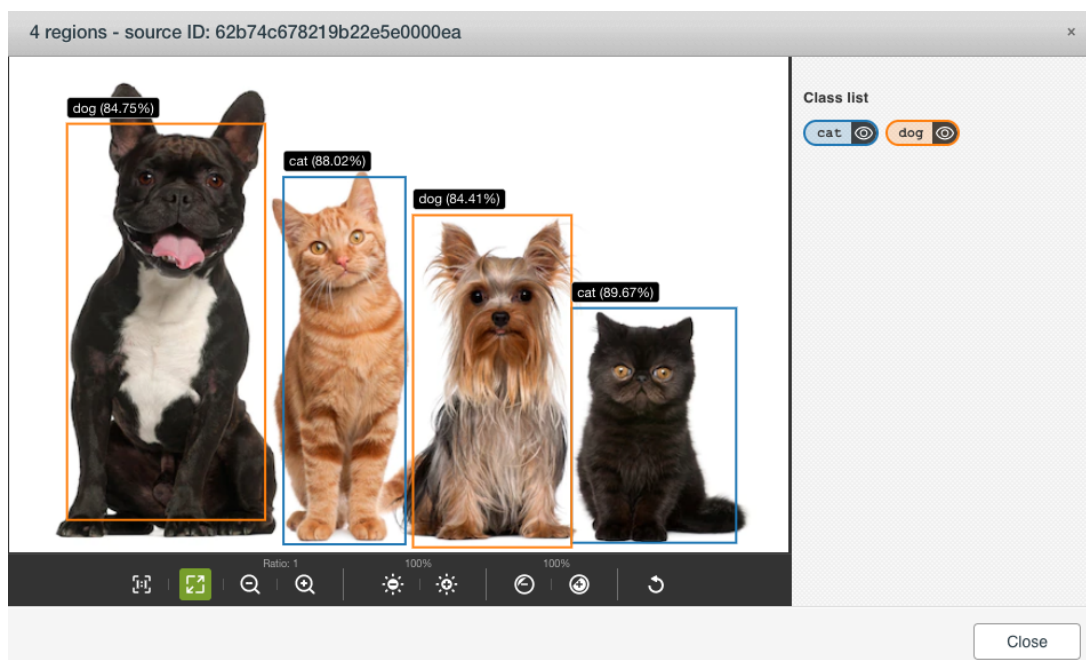


Figure 7.13: Closeup view of an image in a prediction

Keep in mind that this is a new image without annotation, so there are no ground truth regions to compare with.

8. The score threshold slider can be used to filter the predictions. For instance if the threshold is increased to 85%, there will be only two objects predicted in fig:od-pred-result-filtered. The image can be predicted again with the new threshold.

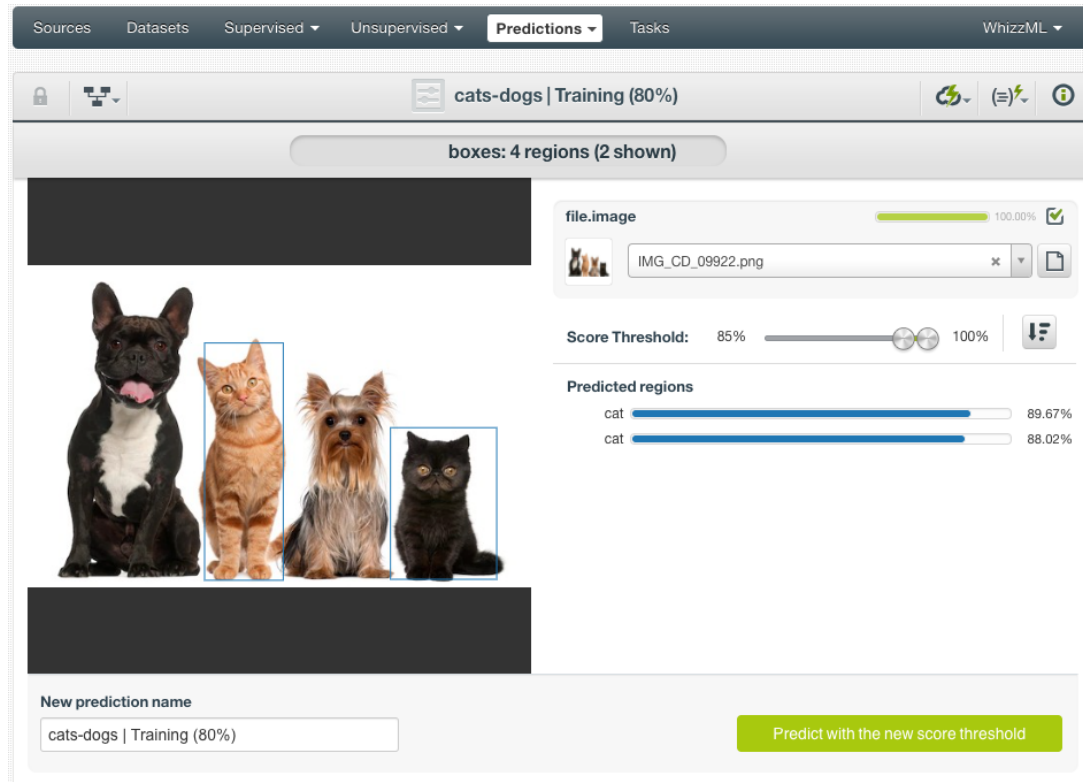


Figure 7.14: Prediction results filtered by a new score threshold

## 7.2.2 Batch Predictions

BigML Batch Predictions allow you to make simultaneous predictions for multiple instances. All you need is the object detection deepnet you want to use to make predictions and a dataset containing the images you want to predict. BigML will create a prediction for each instance in the dataset.

Follow these steps to create a batch prediction:

1. While in the object detection visualization page, click BATCH PREDICTION in the **1-click action menu**, as shown in Figure 7.15.

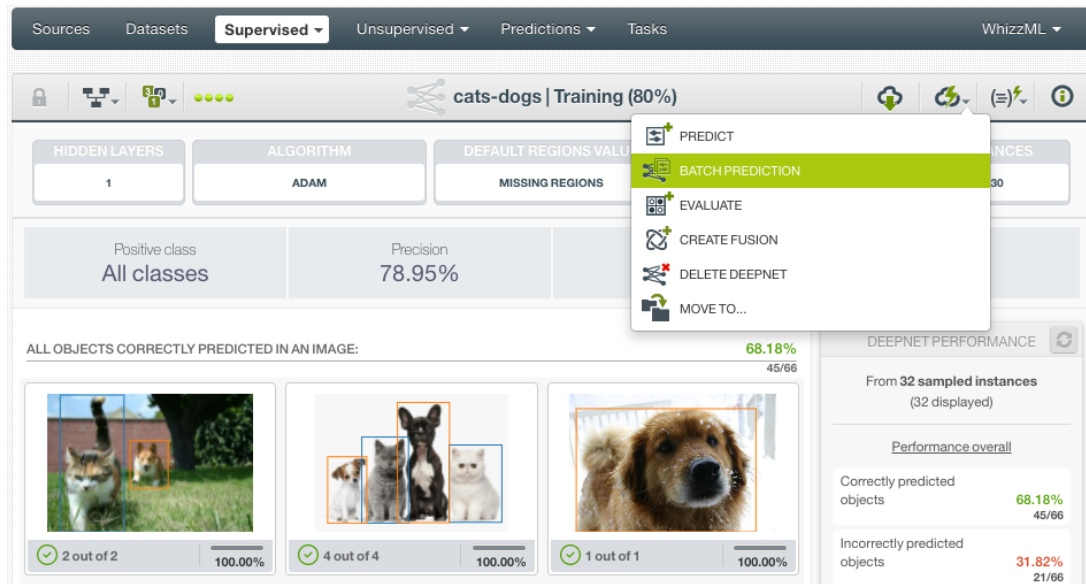


Figure 7.15: Batch predict using the 1-click action menu

Alternatively, click CREATE BATCH PREDICTION in the **pop-up menu** in the deepnet list view. See Figure 7.16.

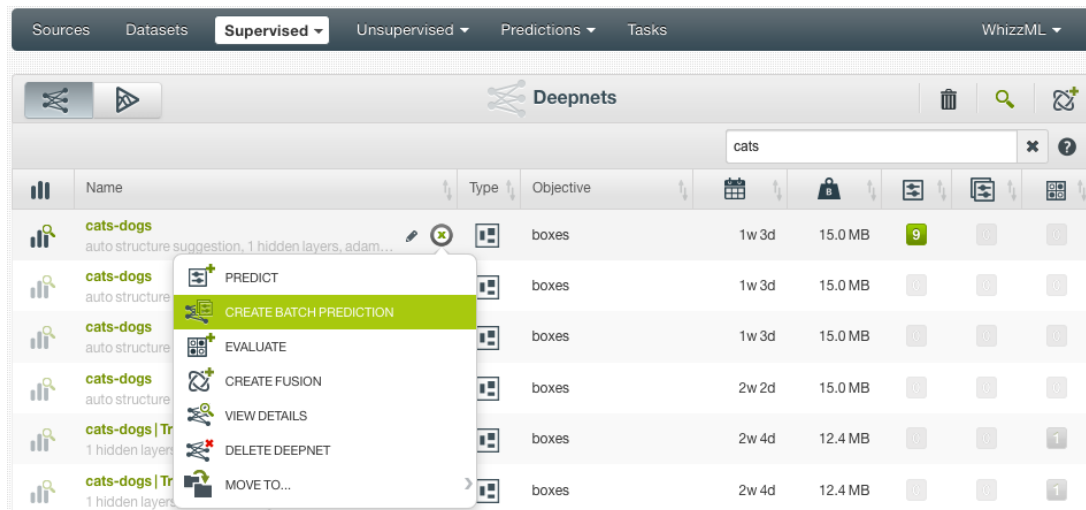


Figure 7.16: Batch predict using the pop-up menu

2. Either option will lead to the **New Batch Prediction** interface, as shown in Figure 7.17. Select the dataset from the dataset input box. Type in the dataset name to filter results. This is the dataset that contains all the images intended for the batch prediction. The dataset was created from a composite source. You can also select a different object detection deepnet from the input selector on the left.

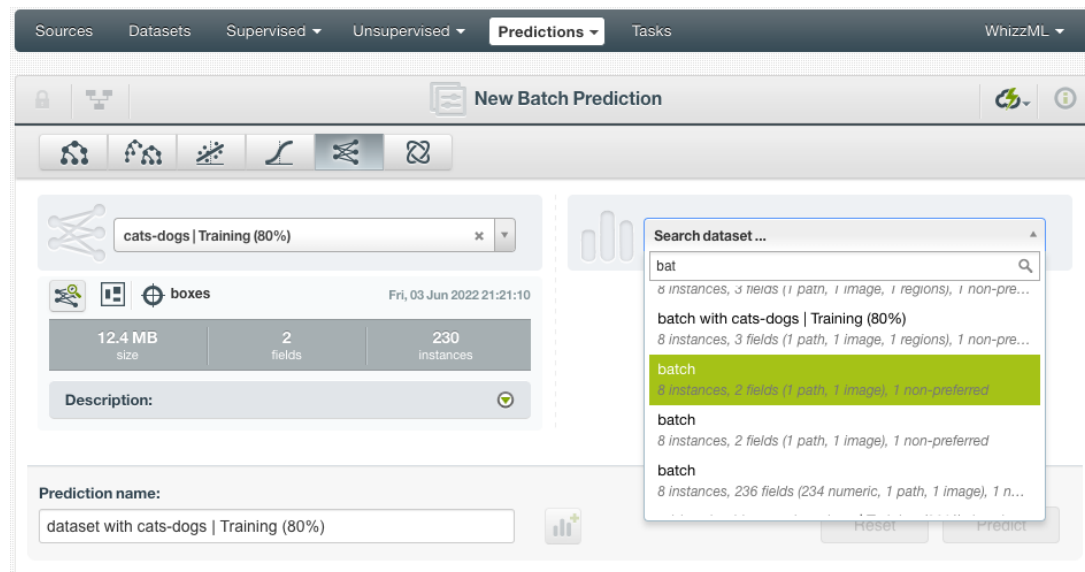


Figure 7.17: New batch prediction

- After the object detection deepnet and the dataset are selected, the batch prediction **configuration options** will appear along with a **preview** of the prediction output (a CSV file), shown in Figure 7.18. The default output format includes all fields from the input dataset and adds an extra column which is a regions field and will contain predicted regions.

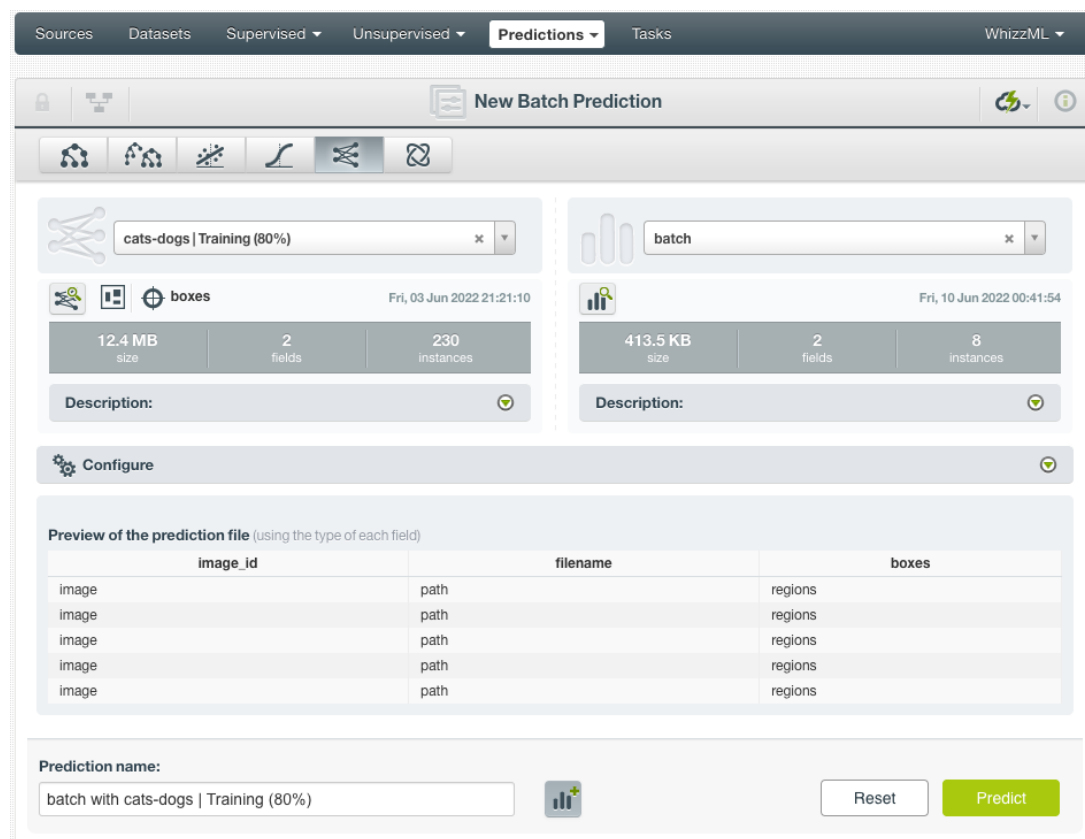


Figure 7.18: Batch prediction configuration options

- Sometimes, the fields in the object detection deepnet and the fields in the input dataset don't match. When this happens, use the **Fields mapping** panel in the configuration options to match

them. In [Figure 7.19](#), the image fields in the deepnet and the dataset have different names. Select the correct ones in both columns of **Deepnet fields** and **Dataset fields** to match them. Also, you can click on the green switcher to set the automatic field mapping by name or by field ID. Moreover, you can search for fields and remove fields from the **Dataset fields** column so they are excluded from the batch prediction.

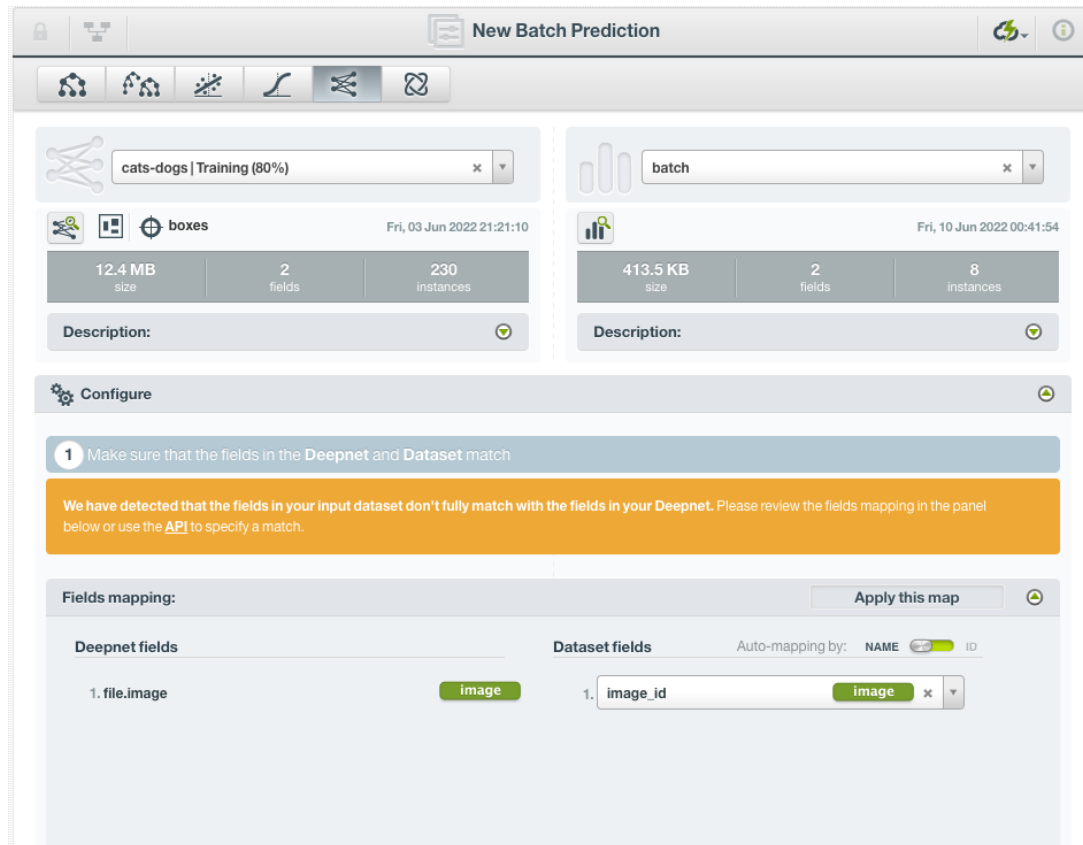


Figure 7.19: Fields mapping

- Below the **Fields mapping** panel is the **Output settings**, shown in [Figure 7.20](#).

**Fields mapping:** Apply this map

**2 [OPTIONAL] Customize prediction output settings**

**Output settings**

**Separator:** (comma) **New line:** Unix, Linux or OS X (LF)

**Prediction column name:** a,b,c predicted

**Output Fields:** image\_id image filename path

**Preview of the prediction file** (using the type of each field)

image_id	filename	predicted
image	path	regions
image	path	regions
image	path	regions
image	path	regions
image	path	regions

**Prediction name:** batch with cats-dogs | Training (80%) Reset Predict

Figure 7.20: Output settings

This is used to configure the output CSV file. The following settings are available:

- **Separator:** this option allows you to choose the best separator for your output file columns. The default separator is the comma. You can also select the semicolon, the tab, or the space.
  - **New line:** this option allows you to set the new line character to use as the line break in the generated CSV file: LF, CRLF.
  - **Output fields:** by clicking on the list icon next to the separator selector, you can include or exclude all input dataset fields from the output file. You can also individually select the fields you want to include or exclude using the multiple output fields selector. **Note: a maximum of 100 fields can be displayed in this selector, but all your dataset fields will be included in the output file by default unless you exclude them.**
  - **Headers:** this option includes or excludes a first row in the output file (and in the output dataset) with the names of each column (input field names, prediction column name, probability column name, etc.). By default, BigML includes the headers.
  - **Prediction column name:** customize the name for prediction column. By default, BigML uses the name of the object detection deepnet's objective field. It's changed to `predicted` in Figure 7.20.
6. By default, BigML generates an output dataset with the new batch prediction that you can later find in the dataset list view on the BigML Dashboard. This option is set by default but you can deselect it by clicking on the icon shown in Figure 7.21.



**New Batch Prediction**

**cats-dogs | Training (80%)**

**batch**

**12.4 MB** size, **2** fields, **230** instances

**413.5 KB** size, **2** fields, **8** instances

**Configure**

**Preview of the prediction file (using the type of each field)**

image_id	filename	predicted
image	path	regions
image	path	regions
image	path	regions
image	path	regions
image	path	regions

**Prediction name:**

batch with cats-dogs | Training (80%)

**Predict**

Figure 7.21: Batch prediction output dataset option

- After the configuration of the batch prediction is finished, click on the green button **Predict** to create the new batch prediction.
- After the new batch prediction is created, it shows that there are two preview panels in [Figure 7.22](#).

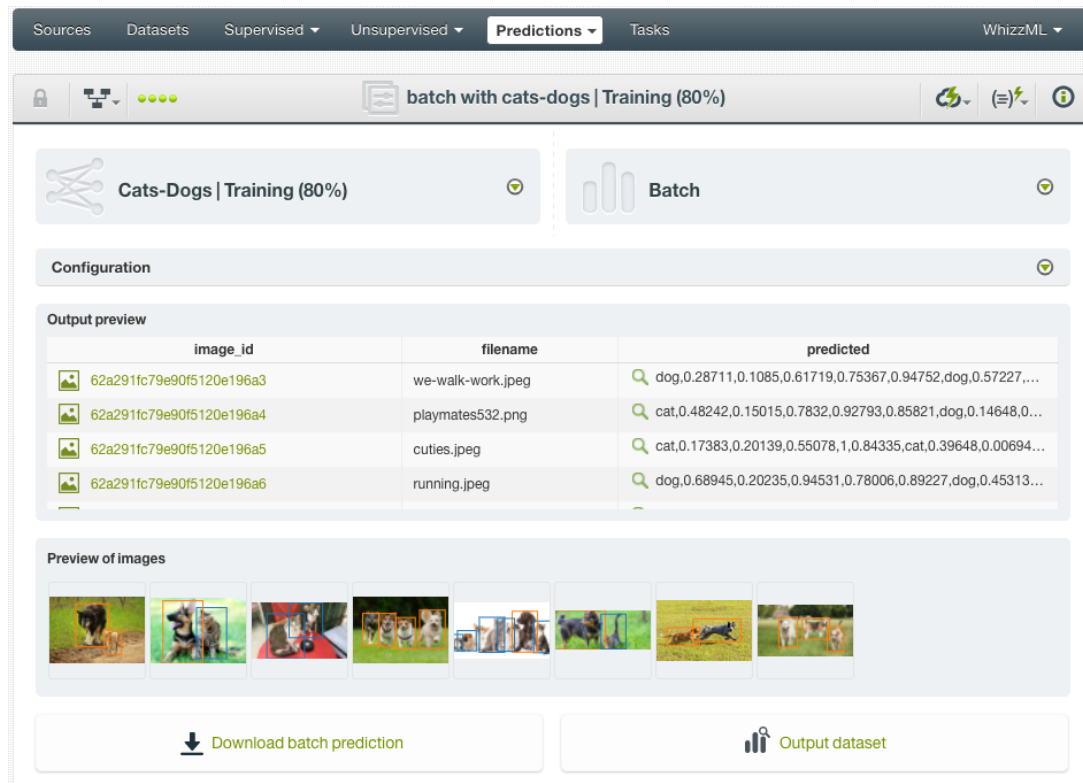


Figure 7.22: A batch prediction

- The **Output preview** is a preview of the output CSV file. It contains sample rows from the CSV and includes all its fields.
- The **Image preview** contains sampled images with predicted regions from the input dataset. It may include all images if the number of the images in the dataset is small.

Clicking on an image will bring a closeup view of the image along with the prediction results, as shown in [Figure 7.13](#). We can click on the class labels on the right to hide or show certain classes.

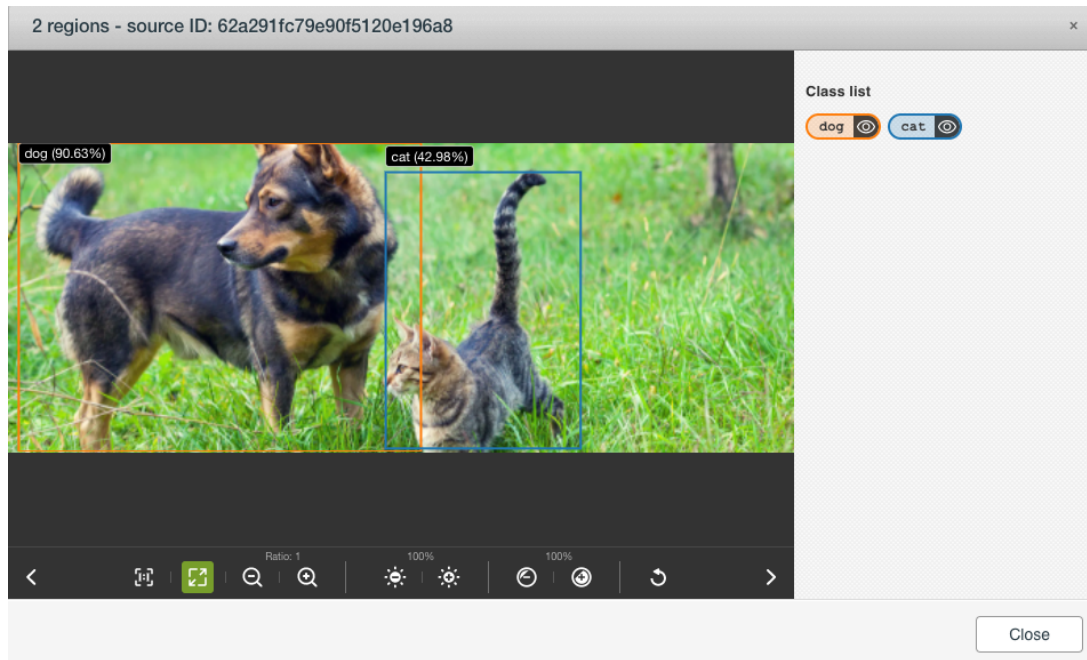


Figure 7.23: Closeup view of an image in a batch prediction

Note that the images in the input dataset were not used for the object detection deepnet training, so there are no ground truth regions to compare with.

9. Figure 7.22 also shows two buttons below the image preview panel. On the left is the button to download the output CSV file. It contains all input dataset instances along with predicted regions for each of them.
10. On the right is the button to access the output dataset. If the dataset option in 6 is deselected, this button won't be available.

Please use the `noidx` option in the `documentclass` invocation.

# List of Figures

1.1	Object detection icon . . . . .	1
1.2	Object detection deepnets in deepnet list view . . . . .	2
2.1	Training a classification deepnet . . . . .	4
2.2	Also training a classification deepnet . . . . .	4
2.3	Training an object detection deepnet . . . . .	5
2.4	Also training an object detection deepnet . . . . .	5
2.5	A training image divided by grids . . . . .	6
2.6	A training image divided by grids . . . . .	7
2.7	A grid cell with the center of a bounding box . . . . .	7
2.8	Another grid cell with the center of a bounding box . . . . .	8
2.9	IOU - Intersection over Union . . . . .	9
3.1	Pixel coordinates . . . . .	13
3.2	Normalized coordinates . . . . .	13
3.3	Converting a <code>table+image</code> composite to an editable composite . . . . .	16
3.4	Add a regions field . . . . .	17
3.5	The annotation interface . . . . .	17
3.6	The adjustment panel . . . . .	18
3.7	An empty Control Panel and a two-class Control Panel . . . . .	18
3.8	The checkbox for marking no regions in the image . . . . .	19
3.9	The character mode . . . . .	20
3.10	Warning for an unannotated status . . . . .	21
3.11	An image composite source . . . . .	21
3.12	Add a regions field . . . . .	22
3.13	The annotation interface . . . . .	22
3.14	Starting drawing a bounding box . . . . .	23
3.15	Finished drawing a bounding box . . . . .	23
3.16	Editing regions . . . . .	24
3.17	Annotation of an image finished . . . . .	24
3.18	Cropping images by regions . . . . .	25
3.19	Configuration options of cropping . . . . .	26
3.20	Cropped images in the output composite source . . . . .	27
4.1	Dataset with a regions field . . . . .	28
4.2	Create deepnet from 1-click action menu . . . . .	29
4.3	Create deepnet from popup menu . . . . .	29
4.4	Create deepnet by configuring options . . . . .	30
4.5	Deepnet configuration options . . . . .	30
4.6	Default regions value . . . . .	31
4.7	Set the training duration for a deepnet . . . . .	31
5.1	Object detection visualization page . . . . .	34

5.2	Object detection visualization layout	35
5.3	Object detection visualization performance panel	36
5.4	Object detection visualization image results	37
5.5	Closeup view of an image	38
5.6	Viewing controls in the image closeup view	39
5.7	Object detection visualization class list	40
5.8	Object detection visualization showing one class	41
5.9	Object detection deepnet visualization	42
5.10	Correctly predicted regions	43
5.11	Comparing correctly predicted regions with their ground truth	43
5.12	Showing the correctly predicted regions	44
5.13	Comparing correctly predicted regions with its ground truth	44
5.14	Showing all ground truth regions	45
6.1	1-click random split of a dataset	46
6.2	Configuring a random split of a dataset	47
6.3	Evaluation list view	47
6.4	Object detection evaluation icon	47
6.5	Creating an evaluation from an object detection deepnet	48
6.6	Creating an evaluation from the deepnet list view	48
6.7	Creating a new evaluation	49
6.8	Evaluation main control	50
6.9	Class detail view	51
6.10	PR AUC: Area Under the Curve of a PR curve	52
6.11	PR AUCH: Area Under the Convex Hull of a PR curve	53
6.12	Closeup view of an image in Image Preview Panel	54
6.13	Summary view	54
7.1	Menu options under Predictions	56
7.2	Prediction list view	57
7.3	Object detection prediction icon	57
7.4	Single prediction icon	57
7.5	Batch prediction icon	57
7.6	Predict using the 1-click action menu	58
7.7	Predict using the pop-up menu	58
7.8	Select a single image source	59
7.9	List the components of a composite source	60
7.10	Select a component of a composite source	60
7.11	Image selected for a single prediction	61
7.12	A single prediction	62
7.13	Closeup view of an image in a prediction	62
7.14	Prediction results filtered by a new score threshold	63
7.15	Batch predict using the 1-click action menu	64
7.16	Batch predict using the pop-up menu	64
7.17	New batch prediction	65
7.18	Batch prediction configuration options	65
7.19	Fields mapping	66
7.20	Output settings	67
7.21	Batch prediction output dataset option	68
7.22	A batch prediction	69
7.23	Closeup view of an image in a batch prediction	70

# Glossary

**Dashboard** The BigML web-based interface that helps you privately navigate, visualize, and interact with your modeling resources. [ii](#)

**Dataset** the structured version of a BigML source. It is used as input to build your predictive models. For each field in your dataset a number of basic statistics (min, max, mean, etc.) are parsed and produced as output. [46](#)

**Objective Field** the field that a regression or classification model will predict (also known as target). [30](#), [46](#)

**Supervised learning** a type of Machine Learning problem in which each instance of the data has a label. The label for each instance is provided in the training data, and a supervised Machine Learning algorithm learns a function or model that will predict the label given all other features in the data. The function can then be applied to data unseen during training to predict the label for unlabeled instances. [ii](#)

## References

- [1] The BigML Team. *Association Discovery with the BigML Dashboard*. Tech. rep. BigML, Inc., Dec. 2015.
- [2] The BigML Team. *Classification and Regression with the BigML Dashboard*. Tech. rep. BigML, Inc., May 2016.
- [3] The BigML Team. *Cluster Analysis with the BigML Dashboard*. Tech. rep. BigML, Inc., May 2016.
- [4] The BigML Team. *Datasets with the BigML Dashboard*. Tech. rep. BigML, Inc., Jan. 2016.
- [5] The BigML Team. *Sources with the BigML Dashboard*. Tech. rep. BigML, Inc., Jan. 2016.
- [6] The BigML Team. *Topic Models with the BigML Dashboard*. Tech. rep. BigML, Inc., Nov. 2016.



