Kingdom of Saudi Arabia
King Saud University
College of Computer & Information Sciences
Department of Computer Science

College of Computer and
Information Sciences

"Visual Relationship Detection in Image Processing"

By:

Feras Aloudah 435104523
Yasser Alrezehi 436104136
Fares Abanmy 436108121
Abdulrhman Al-Qinyah 437103485

Under the supervision of:

Dr. Said Kerrache

April 30, 2020

ACKNOWLEDGMENTS

We would like to express our very great appreciation to Dr. Said Kerrache for his valuable and constructive suggestions during the planning and development of this research work. His willingness to give his time so generously has been very much appreciated.

ABSTRACT

In the research area of computer vision and artificial intelligence, learning the relationships of objects is an important way to deeply understand images. Formally, the task consists in generating a set of relationships represented as subject-predicate-object triples that describe the image content. Most of the recent works detect visual relationships by learning objects and predicates respectively in feature level, but the dependencies between objects and predicates have not been fully considered. Our visual relationship detection aims to describe the interaction between two or more objects. Instead of building one model for relationship detection, we build/use multiple models working together to produce relationships. This is less expensive as it reduces the task complexity and increases the detection performance on large scale data. Indeed, the labeling needed to detect relationships on data is too difficult due to the required human effort and a large number of possible relationships. This makes it hard to explore all relationships based only on the visual appearance of objects. Furthermore, the annotations for visual relationships are usually incomplete which increases the difficulty of model training and evaluation. In this project, we propose a method for visual relationship detection that uses this insight to train the visual model for detecting two objects that compose an object-pair and classify them as either determinate or un-determined relationship and another model to predict the relationship using information from the first model. Both models work individually and later the final relationships are decided according to the score from the two models to produce multiple relationships per image.

ARABIC ABSTRACT

في مجال البحث الخاص برؤية الكمبيوتر والذكاء الاصطناعي، يعد تعلم علاقات الكائنات طريقة مهمة لفهم الصور بعمق من خلال إنتاج علاقة ممثلة على أنها ثلاثية العناصر الموضوعية. تكتشف معظم الأعمال الحديثة العلاقات المرئية عن طريق تعلم الأشياء والتنبؤات على التوالي في مستوى الميزة ، لكن التبعيات بين الكائنات والمسندات لم يتم أخذها في الاعتبار بشكل كامل. يهدف اكتشاف العلاقة المرئية إلى وصف التفاعل بين كائنين أو أكثر. يختلف عن بناء نموذج واحد لاكتشاف العلاقة، بل بناء نماذج متعددة تعمل معاًا لإنتاج علاقات أقل تكلفة وتقلل التعقيد وتزيد من أداء الكشف على نطاق واسع لأن وضع العلامات اللازمة لتمييز العلاقات على البيانات أمر صعب للغاية بسبب الجهد الإنساني المحدود وعدد العلاقات المحتملة أكبر بكثير مما يجعل من الصعب استكشافه فقط بناءً على المظهر المرئي للكائنات ، وعادةً ما تكون التعليقات التوضيحية للعلاقات البصرية غير مكتملة مما يزيد من صعوبة التدريب النموذجي والتقييم. في هذا المشروع ، نقترح نموذجاًا يسمى ْاسم المشروعْ لاكتشاف العلاقة المرئية. يستخدم هذا البصيرة لتدريب النموذج البصري للكشف عن كائنين يؤلفان زوجاًا من الكائنات ويصنفانهما على أنهما علاقة محددة أو غير محددة ، ونموذج آخر يتنبأ بالعلاقة باستخدام معلومات من النموذج الأول يعمل كلا النموذجين بشكل فردي ثم لاحقاًا يتم تحديد العلاقات النهائية وفقاًا للنتيجة من النموذجين لإنتاج علاقات متعددة لكل صورة

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The computer vision field has been growing so fast in the last few years, its applications range from basic object detection in images to predicting cancer and are practically endless.
Our project focuses more on object relationships in images, and how to detect them and classify them accordingly. For example, an image with a person and a bicycle might involve the man riding, pushing, or even falling off of said bicycle.

Figure 1.1: Even though all the images contain the same objects (a person and a bicycle), it's the relationship between the objects that determine the end goal.

–We will be using labeled images from Open Images an open-source dataset of approximately 9 million images that have been annotated with image-level labels and bounding boxes spanning thousands of classes. We use these labels and classes to detect multiple types of relationships.

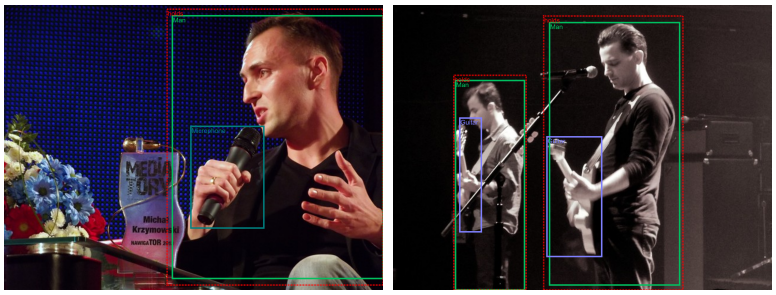

Figure 1.2: Given an image as input, we detect multiple relationships and then we draw bounding boxes around the two objects explaining the relation between them.

1.1 Problem statement

Nowadays image classification is one of the most popular subjects in machine learning, which involves a lot of steps to output an accurate result. Our paper focuses on detecting relationships in images, and how to classify these relationships accordingly.

We will be using an open-source dataset called Open Images [7], which consists of a large number of labeled images that we can use to help us when we train our models.

Some of the challenges that we might face are, cleaning the data and needing a lot of training time since visual relationship detection needs to locate and catch a lot of semantic information which takes a lot of time.

1.2 Goals and Objectives

The goal of this research is to classify images by detecting the visual relationships from the objects that are detected in the image.

The objectives of the proposed approach are as follows:

1. Analyze the fully supervised set of images with relationship annotations, fully labeled.
2. Use these images to train on object detection.
3. Use the same images to train on relationship detection.

1.3 Research Scope

What will be covered in this research:

- Extracting certain features from an image to detect objects.
- Use the detected object's features to detect relationships.

What will not be covered in this research:

- Realtime relationship detection that works with videos.
- Detecting relationships that are outside of our dataset's scope.

1.3.1 The Dataset

The whole dataset consists of 9,011,219 images for the training set, 41,620 images for the validation set, and 125,436 images for the test set. These images are annotated with image-level labels (The image as a whole is annotated with meaning. Example: "A picture of a cat sleeping"), object-bounding boxes, object segmentation masks, and visual relationships. We're only interested in the bounding boxes, and visual relationships so we won't be using the whole dataset [7].

Chapter 2

Background

2.1 Machine Learning

Deep Learning is but a small part of Machine Learning. So, before we dive into Deep learning, we first need to know more about Machine Learning and how it works. If computers could learn, think or even act like human beings, how big of a leap would science jump. [1] Teaching a machine and making it think is the basic idea of Artificial Intelligence. When programmable computers were designed for the first time, people started wondering whether computers will become intelligent, more than a hundred years before one was built [2].

Today, Artificial Intelligence is a field full of practical applications and active research topics. We see intelligent software as a way that automates routine work, understand speech or images, make diagnoses in medicine and support basic scientific research.

Recently, Artificial Intelligence has been beneficial by solving problems that are difficult for humans beings, but relatively straightforward for computers.

"The true challenge to artificial intelligence proved to be solving the tasks that are easy for people to perform but hard for people to describe formally, problems that we solve intuitively, that feel automatic, like recognizing spoken words or faces in images" [2].

After we understood Artificial Intelligence, we can dive into Machine Learning because after all, Machine Learning is a specific part of AI. Machine learning is the system's way of understanding problems by looking at past events and recognizing patterns.

Although Machine learning has many techniques, it can be categorized into three general types:

1. **Supervised Learning:** this technique focuses on giving labeled data to discover a rule that enables the computer to recreate the outputs or map the inputs to certain outputs. For example, to teach an algorithm to distinguish between dogs, and cats

you need to give your algorithm thousand of images of each animal, with each image being labeled by either “dog”, or “cat”,after that the algorithm would be able to classify new images with a minimum error percentage by recognizing patterns in each image [4].

2. **Unsupervised Learning:** unlike supervised learning, the data given is not labeled, which means that the algorithm needs to find structure and patterns. Without the images having a label that identifies the output.

3. **Semi-Supervised Learning:** is the combination of the previous learning styles, with means we give it labeled and unlabeled inputs, which means the algorithm needs to find patterns from the entered inputs but with little help.

2.1.1 Gradient Boosting

Gradient boosting machines are a kind of decision tree algorithm that works by building an ensemble shallow trees in sequence with each tree being able to learn and improve from the previous. While shallow trees are weak at making predictions, they can be boosted and tuned to produce very accurate predictions.

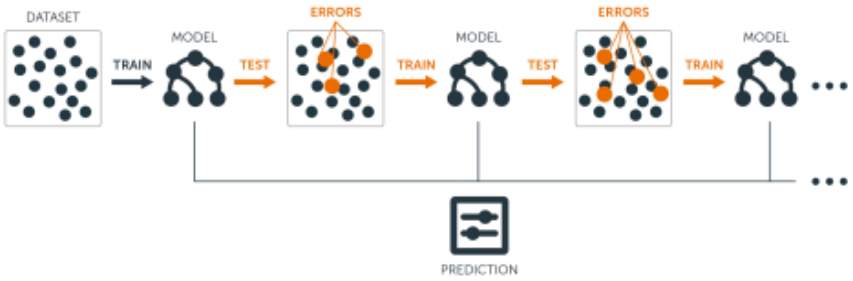

Figure 2.1: Sequential ensemble approach. [15]

Boosting requires base learners of weak models that have an error rate that is only slightly higher than randomly guessing, and the idea is that each model in the sequence improves slightly by focusing on the parts the previous model was having residuals with. [15]

1. Fit a decision tree to the data:
$$F_1(x) = y$$

2. We then fit the next decision tree to the residuals of the previous:
$$h_1(x) = y - F_1(x)$$

3. Add this new tree to our algorithm:

$$F_2(x) = F_1(x) + h_1(x)$$

4. Fit the next decision tree to the residuals of:

$$F_2 : h_2(x) = y - F_2(x)$$

5. Add this new tree to our algorithm:

$$F_3(x) = F_3(x) + h_3(x)$$

6. Continue this process until some mechanism (i.e. cross validation) tells us to stop.

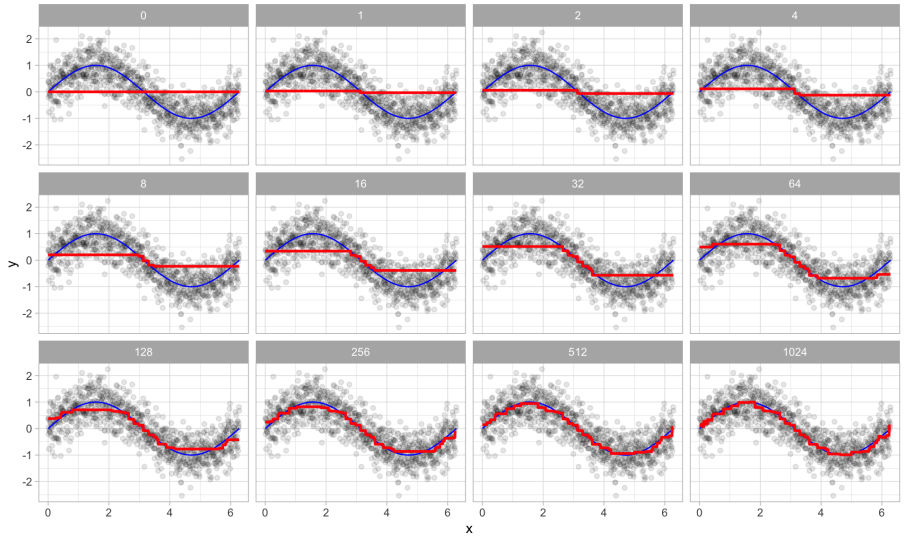

Figure 2.2: A simple example showing how the model predictions improve over multiple trees. where a single predictor (x) has a true underlying sine wave relationship (blue line) with y along with some irreducible error. [15]

2.2 Deep Learning

Machine Learning, Deep Learning, and Neural Networks have grown in the last few years and its application spread throw the various majors. Deep Learning is an Artificial Intelligence function that mimics the way human beings' minds works in processing and connects patterns for using it to make a decision. Deep Learning is a part of Machine Learning that has networks known as Neural Network.[5]

2.2.1 Neural Networks

Neural networks aim to imitate the way the human brain work. and its goal is to solve problems the same way we humans do. Neural networks consist of multi-layer networks, that might have a goal of classifying things or make predictions.

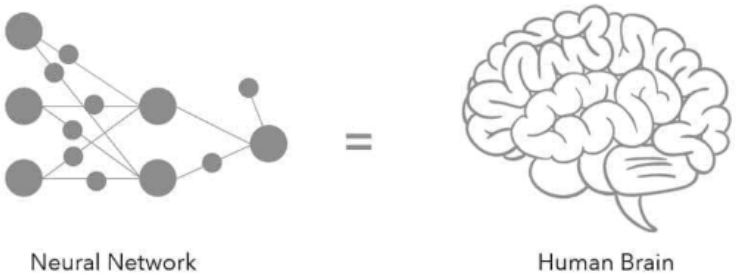

Figure 2.3: A neural network next to a drawing of the human brain. [4]

Layers: The neural network layers consist of many nodes that are used to extract features from our input, these layers consist of three types [4]:

1. Input layer: this layer consists of nodes that are responsible for accepting the input and passing it to the hidden layers.
2. Hidden layers: these layers carry no value before training, but after the model is finished training these layers become responsible for determining what's important in describing the relationship from the observed data.
3. Output layer: this layer is responsible for collecting the output from the hidden layers and then determining what to finally output.

Input moves from one layer to another through a process called forward propagation. This process consists of the summation of all the nodes' values followed by an activation function being applied to them.

Activation functions are used to determine how much a node should contribute to the output or if the node needs to be deactivated. There are many types of activation functions to use with each having pros and cons depending on the situation.

$$\frac{1}{1+e^{-x}}$$

The sigmoid function, for example, is used to transfer the input into a number between zero and one. Where a certain threshold is used to determine the node's contribution.

$$F(x) = Max(0, x)$$

Another example of a function that is widely used in image recognition is the rectified linear unit, there are many theories as to why ReLU performs much better than other functions, but no general agreement has been reached. [16]

To evaluate a network a cost function is needed to calculate the error rate of the network, this function is used to minimize the error rate of a network.

These are some examples of a cost function [4]:

- MSE (**mean square error**):

$$\frac{1}{n}\sum_{i=1}^{n}(Predicted_i - Actual_i)^2$$

- RMSE (**root mean square error**):

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(Predicted_i - Actual_i)^2}$$

2.2.2 Softmax

Softmax is used in machine learning to turn arbitrary values into probabilities, we can think of it as calculating the confidence of our network that the output is correct. Softmaxing helps us transform simple classification problems into probabilities that could make our final decision easier. [22]

Softmax works by first calculating the sum of all the powers of e, followed by dividing each power of e to the sum to get its probability.

$$s(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}$$

2.3 Convolutional Neural Networks

Convolutional neural networks are basically a subclass of regular neural networks that have convolutional layers. These layers are used to capture information much faster while still being accurate. [17]

CNNs have many layers that are needed to build one, some of these are:

- Convolutional
- Activation
- Pooling

- Fully-connected - FC
- Dropout

You can describe a CNN using text diagrams (e.g., Input -> Conv -> Activation -> FC), but only conv and FC layers are changed during training, while the rest have set parameters defined beforehand.

2.3.1 Convolutional Layers

These layers make up the core parts of a CNN, they are the reason why we call them convolutional neural networks. The layers consist of K learnable filters with each having a width and a height and almost always make up a square, these are usually small in size but they extend the full depth of the volume. [17]

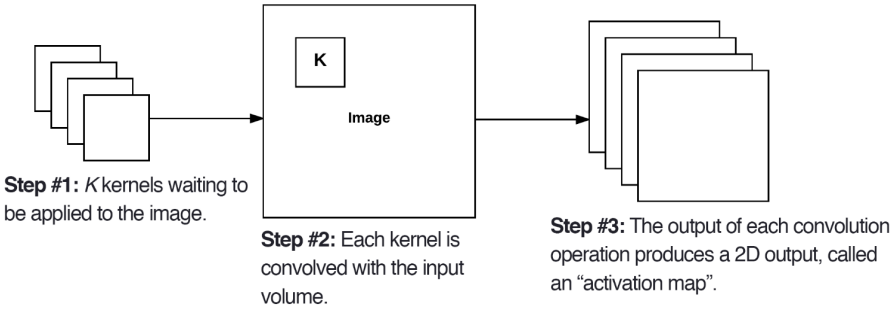

Figure 2.4: Left: At each convolutional layer in a CNN, there are K kernels applied to the input volume. Middle: Each of the K kernels is convolved with the input volume. Right: Each kernel produces a 2D output, called an activation map.[16]

For example, the depth of an image is the number of channels it has (e.g., RGB means 3 channels) and the deeper it goes into the network the depth will be the number of filters that are being applied in the previous layer. After we apply all of the filters we get K 2D activation maps, that we stack to form the final output volume.

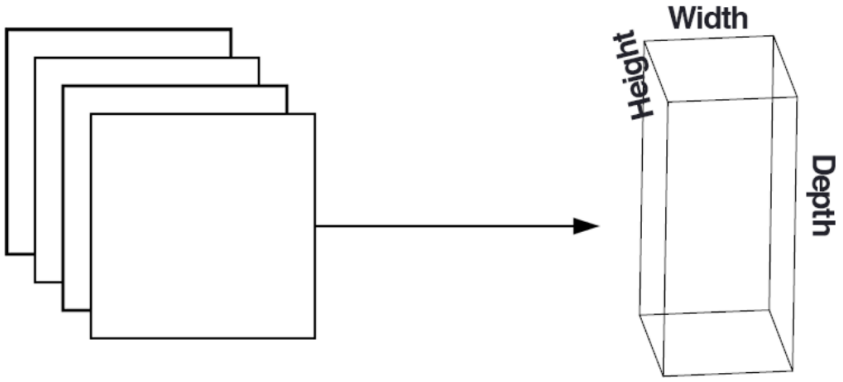

Figure 2.5: After obtaining the K activation maps, they are stacked together to form the input volume to the next layer in the network.[17]

Every activation map contains the output of a small part of the input and depending on these maps the network improve filters to activate only when looking at specific features in these maps. These filters begin looking for filters that are a bit low level and starts looking for high-level features the deeper it goes into the network.

2.3.2 Activation Layers

After each conv layer, we apply an activation function (e.g., ReLU). These layers are not learned since they hold no parameters that require learning and are often not mentioned in a network's diagram since they are always assumed to follow a conv layer.

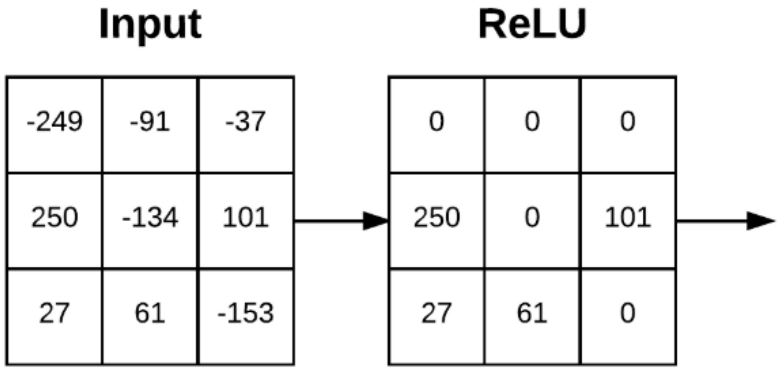

Figure 2.6: An example of an input volume going through a ReLU activation, max(0,x). Activations are done in-place so there is no need to create a separate output volume although it is easy to visualize the flow of the network in this manner. [17]

2.3.3 Pooling Layers

Pooling layers are used to reduce the size of an input volume and usually used between conv layers. These layers work by reducing the size of the input, this results in fewer parameters and faster computation, this also helps in avoiding overfitting the network.

Pooling is applied on each slice of our input, by using either a max or average function. Max pooling is usually applied during the middle of the network while Average pooling is only applied to the final layer of the network if we are trying to avoid using a final connected layer.

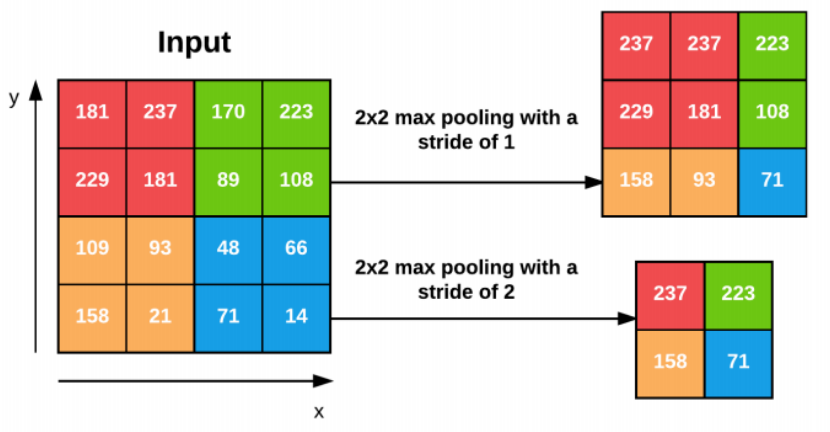

Figure 2.7: Left: Our input 44 volume. Right: Applying 22 max pooling with a stride of S = 1. Bottom: Applying 22 max pooling with S = 2 – this dramatically reduces the spatial dimensions of our input. [17]

2.3.4 Fully-connected Layers

FC layers are used to connect the output of the other layers to help us reach a final decision and are usually used at the end of the network. A network can have multiple FC layers (e.g., one for applying weights to the analyzed features and the other to reach a decision on our output).

FC layers work by taking the output of previous layers and then flattening them and turns them into a single vector that holds values on each feature so it can be used to determine our final decision.

2.3.5 Dropout

Dropout is a form of regularization that has the purpose of preventing overfitting, it works by randomly ignoring some outputs from the layers. This results in some noise when training, but this also results in each layer being updated with a different view of the modified layer.

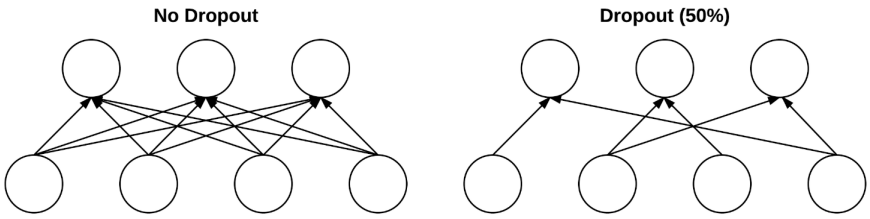

Figure 2.8: Left: Two layers of a neural network that are fully-connected with no dropout. Right: The same two layers after dropping 50% of the connections. [17]

The reason dropout reduces overfitting is because dropout makes sure that no single node is responsible for activation when presented with a certain input, but instead, it relies on multiple nodes that activate when presented with said input.

2.4 Image Processing

Image processing is such a complex subject when it comes to machine learning, and it has jumped a huge leap in the last decade. It has many applications in the real world that range from real-time object detection to help in diagnosing a cancer patient early.

This is a very big subject and we can't cover it in its entirety, so we're just gonna look at some methods that are used to process images that could give us a better picture of what's going on under the hood.

2.4.1 Image Gradient Vector

Image gradient vector is a metric that is used to figure out the color changes of each pixel in both the x-axis and y-axis. This is done by calculating a vector of partial derivatives as follows: [18]

$$\bigtriangledown f(x,y) = \begin{pmatrix} g_x \\ g_y \end{pmatrix} = \begin{pmatrix} \frac{\alpha f}{\alpha x} \\ \frac{\alpha f}{\alpha y} \end{pmatrix} = \begin{pmatrix} f(x+1,y) - f(x-1,y) \\ f(x,y+1) - f(x,y-1) \end{pmatrix}$$

the $\frac{\alpha f}{\alpha x}$ term represents the partial derivative on the x-axis which is calculated by getting the color difference between the pixels on the left and right of our target pixel. Much like the previous term represents the partial derivate on the y-axis, where it is calculated by looking at the pixels above and below our target pixel.

Image gradient vectors consist of two major parts

- Magnitude: which represents the L2-norm of the vector $g = \sqrt{g_x^2 + g_y^2}$
- Direction: which is the arctangent of the ratio between the partial derivatives $\phi = arctan(g_y/g_x)$

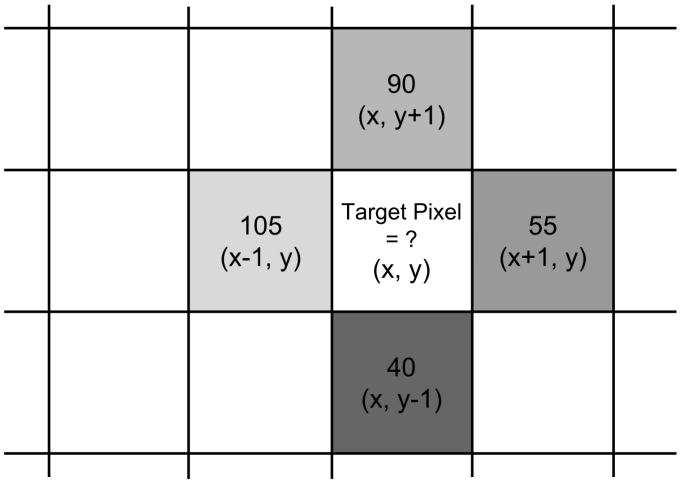

Figure 2.9: To compute the gradient vector of a target pixel at location (x, y), we need to know the colors of its four neighbors. [18]

2.4.2 Histogram of Oriented Gradients

HOG is a way to extract features from pixel colors to help us classify objects in an image, it works by first preprocessing the image (resizing and color normalization), followed by computing the gradient vectors of every pixel and their magnitude and direction.

The image is then divided into many 8x8 pixel cells. The magnitude values of these cells are then grouped into nine buckets of unsigned direction that goes from 0 to 180. This may result in a pixel being between two buckets which then needs to be split between them.

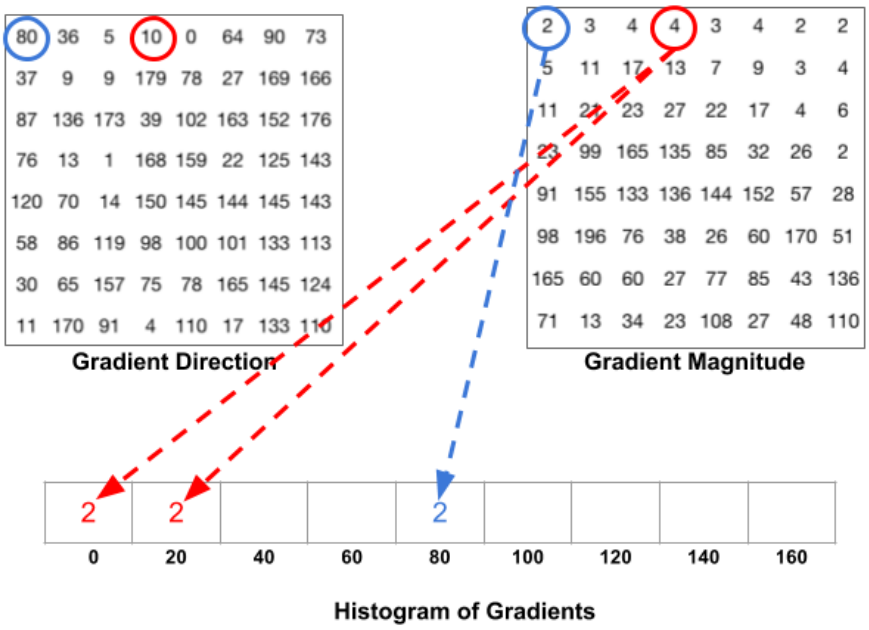

Figure 2.10: How to split one gradient vector's magnitude if its degrees is between two-degree bins. [19]

Lastly, we slide a 2x2 (16x16) cells block through the whole image, and in each block region, four histograms of four cells are concentrated into a one-dimensional vector that has 36 values that can be used to classify an object from an image. [18]

2.4.3 Image Segmentation

Images usually have multiple objects in them, so we need to identify each object's region in an image so that our prediction is accurate. There are many ways to segment an image, one of which is the algorithm that Felzenszwalb and Huttenlocher proposed which is graph-based. [20]

There are many ways to construct a graph out of an image that we can use to apply this algorithm.

- Grid graph: where each pixel is only connected to its neighbors and its weight is the absolute difference between the intensity values of the pixels.
- Nearest neighbor graph: where each pixel is represented in the feature space (x, y, r, g, b) where x and y represents the pixel's location and RGB representing its color. The weight is calculated by getting the euclidean distance between two pixels' feature vectors.

The algorithm follows a bottom-up procedure where edges are sorted by weight in ascending order. Where we take each edge (consisting of vi and vj) and compare the component of the two vertices and see if the difference between them is high enough and maybe merge them if their weight is under a certain threshold. [18]

2.4.4 Selective Search

Selective search is an algorithm that is used to get region proposals that might contain objects in them. The algorithm works by first applying image segmentation to an image to create regions to start with. We then use a greedy algorithm to calculate similarities between all of the nearest neighbors and grouping the most similar regions together. This process is repeated until the whole image becomes a single region.

Algorithm 1: Hierarchical Grouping Algorithm

DontPrintSemicolon **Input**: (colour) image
Output: Set of object location hypotheses L

Obtain initial regions $R = \{r_1, \cdots, r_n\}$ using Felzenszwalb and Huttenlocher (2004) Initialise similarity set $S = \emptyset$;
foreach *Neighbouring region pair* (r_i, r_j) **do**
  Calculate similarity $s(r_i, r_j)$;
  $S = S \cup s(r_i, r_j)$;

while $S \neq \emptyset$ **do**
  Get highest similarity $s(r_i, r_j) = \max(S)$;
  Merge corresponding regions $r_t = r_i \cup r_j$;
  Remove similarities regarding r_i : $S = S \setminus s(r_i, r_*)$;
  Remove similarities regarding r_j : $S = S \setminus s(r_*, r_j)$;
  Calculate similarity set S_t between r_t and its neighbours;
  $S = S \cup S_t$;
  $R = R \cup r_t$;
Extract object location boxes L from all regions in R;

Figure 2.11: The detailed algorithm of Selective Search. [18]

2.4.5 VGG16

VGG16 is a 16 layer CNN and used to be considered very deep in its time. This network passes a 224x224 RGB image through five blocks of conv layers where each layer has an increasing number of 3x3 filters. These blocks are separated by max-pooling layers that are performed over 22 windows. In the end, there are three fully-connected layers that are used to flatten the output and finally, the output is fed into a softmax layer that is used to get our classification. [21]

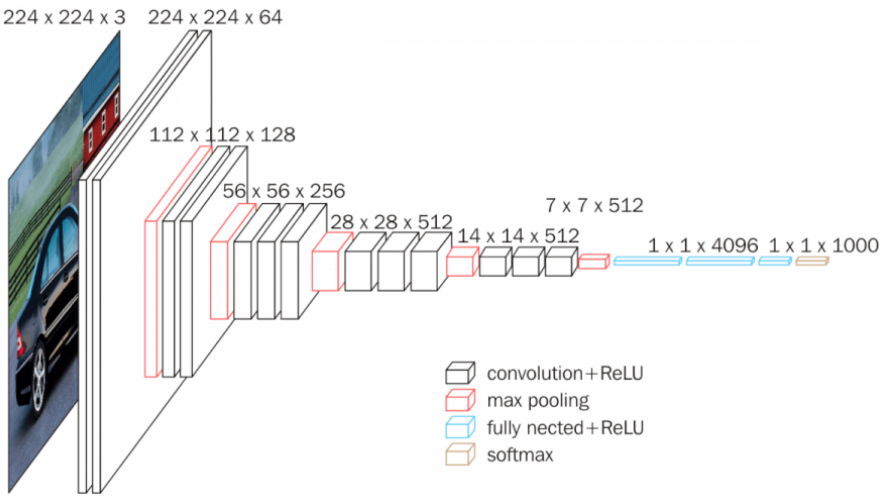

Figure 2.12: The VGG16 architecture.

Chapter 3

Related Work

Visual relationship detection has been a very popular subject in recent years, and there has been a lot of open-source development in the matter [25-27]. The process involves multiple steps that include detecting objects in an image, followed by translating and understanding the relationships between them. We will be first be exploring a lot of related work that strives to improve object detection and then we'll at work that focused more on relationship processing.

3.1 Deep Learning for Object Detection

Object detection is a vital part of detecting relationships since the whole process depends on it, so naturally, there has been an increase in research regarding the subject. In this section, we will focus on the different ways to detect objects in an efficient manner.
The most used method is CNN and early approaches utilize sliding windows along with it [28-29]. But this becomes a problem when the network becomes very large and hard to control which results in lower accuracy overall.

3.1.1 R-CNN

This continued for a while until R-CNN was introduced (Girshick et al., 2014) [9], which is short for "Region-based Convolutional Neural Networks", it works by first doing a selective search that provides region proposals that could potentially contain objects which are called "regions of interest" (RoI) which are much more manageable than the sliding window method, and then it extracts CNN features from each region to classify.

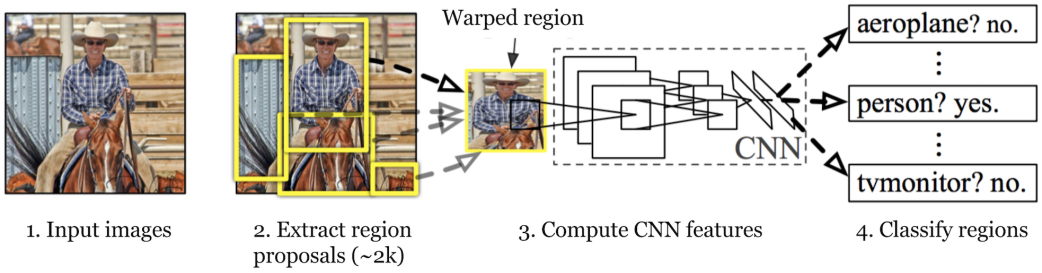

Figure 3.1: R-CNN architecture [9]

3.1.2 Fast R-CNN

Girshick reintroduced a better version of R-CNN in 2015 [10], that worked by inputting the whole image instead of the RoI to generate a convolutional feature map, which is then used to identify the region proposals, which is followed by RoI pooling and finally using a softmax layer to make our prediction.

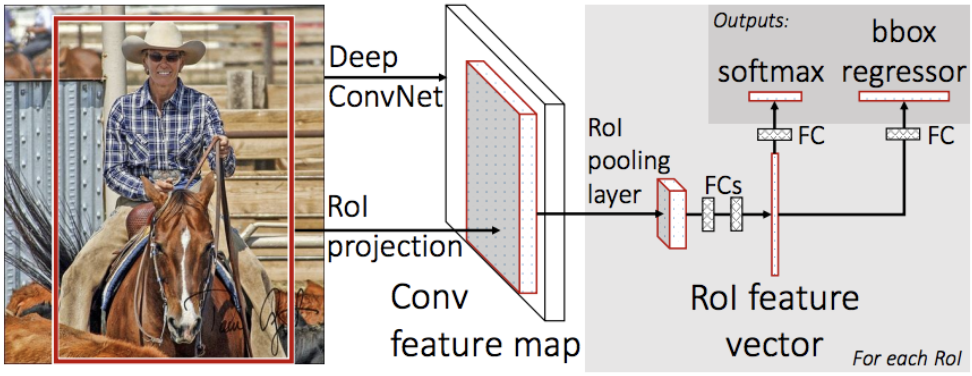

Figure 3.2: Fast R-CNN architecture [10].

3.1.3 Faster R-CNN

Both R-CNN and Fast R-CNN use selective search to find the region proposals, which is very slow and lowers the performance a lot. Which is why (Ren et al., 2016) [11] introduced an object detection algorithm that dropped selective search and made the network learn the region proposals. Faster R-CNN works similarly to Fast R-CNN, but instead of using selective search on the feature map, a separate network is used to predict the region proposals. This makes it really fast that it can be used on real-time object detection.

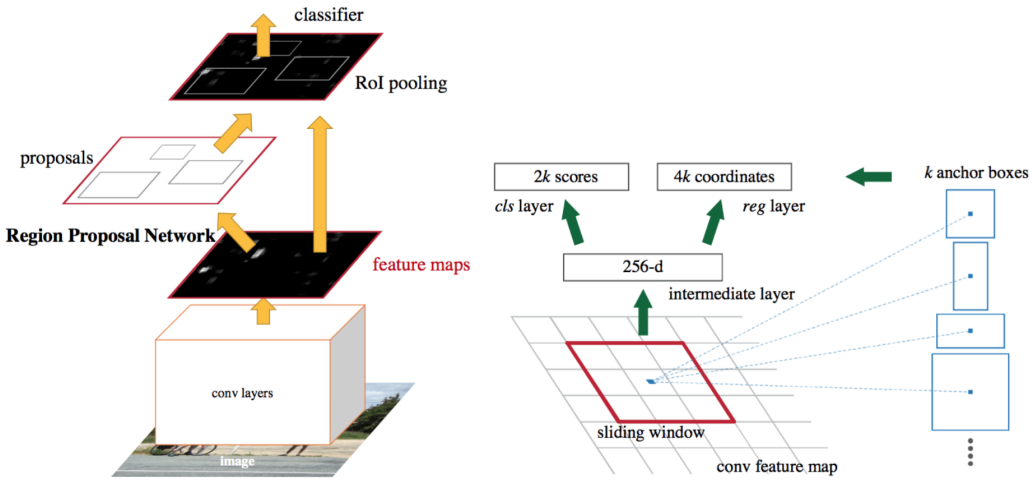

Figure 3.3: Faster R-CNN architecture [11].

3.1.4 YOLO

While the above methods work really well "You Only Look Once" or YOLO for short was proposed (Redmon et al., 2015) [6], which is much faster but in return, it's less accurate than R-CNN based methods. YOLO works by splitting an image into an SxS grid and then predict bounding boxes and class probabilities for each region of the grid. The bounding boxes with a high enough class probability are then selected and used to locate the object within the image. Other versions of YOLO have been introduced [12] that significantly improved the algorithm which makes YOLO a great option for real-time object detection since it runs really fast.

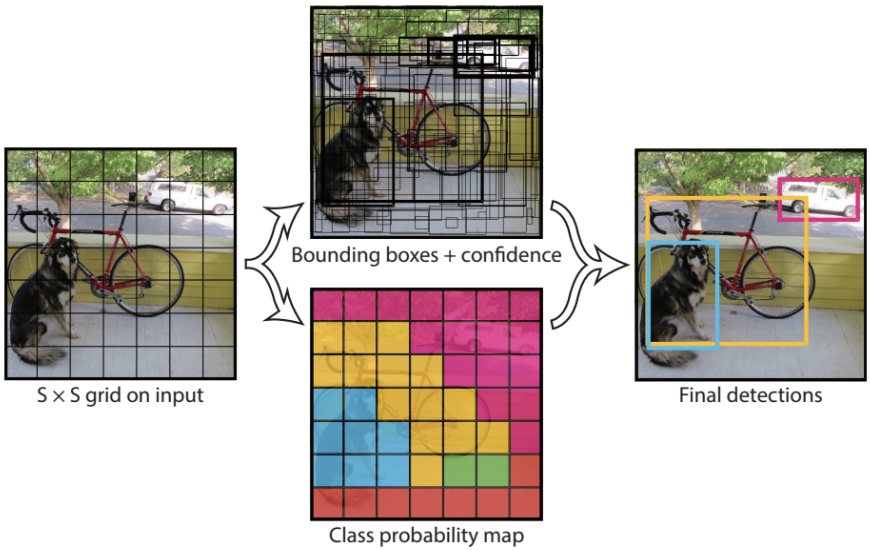

Figure 3.4: YOLO model process [6].

3.2 Relationship Detection

After we finish detecting the objects in an image, we move on to detect the relationship that unites them together. We can define relationships in a number of ways, such as spatial relationships (e.g., "on", "under", or "inside of"), interactive relationships (e.g., "human playing a guitar", or "human holding a water bottle), or "is" relationships (e.g., "chair is wooden", or "plastic water bottle").

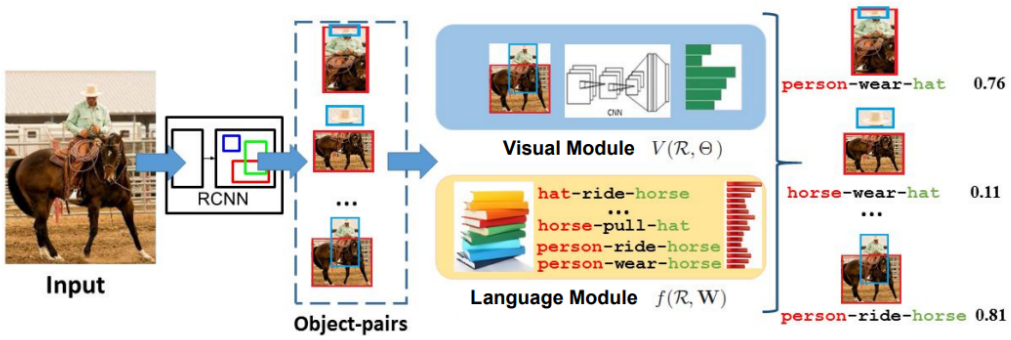

Figure 3.5: An overview of the visual relationships detection pipeline used in [13], where the first part is to detect the objects in an image, and then pair objects together depending on each object offset, followed by inputting these pairs to the Visual Module, and Language Module to score each relationship.

There has been a lot of research regarding what's the best method to detect relationships, Lu et al., (2016) [13] proposed detecting these relationships by using a language module (see Figure 3.5) to try and generate all possible relationships between the objects, they noticed that these relationships are semantically related together (e.g., "man riding a horse" is the same as "man riding an elephant"), since you can infer this relationship from past learned relationships. As seen in Figure 3.5, The visual module is responsible for detecting the objects, and their predicates, while the language module works by concatenating the two detected objects into a relationship using a pre-trained model to generate the relationship and a projection function to score the relationship.

$$f\left(R\left\langle i,k,j\right\rangle ,W\right)=W_{k}^{T}\left[word2vec\left(t_{i}\right),word2vec\left(t_{j}\right)+b_{k}\right]$$

"where R<i, k, j> represents the relationship between i, and j. tj is the word (in text) of the jth object category. wk is a 600 dim. vector and bk is a bias term. W is the set of w1, b1, ... , wk, bk, where each row presents one of our K predicates." [13].

While the above solution focuses on spatial, and verb relationships. Girshick et al., (2018) [14] focused more on human-centric relationships (e.g., "man cutting a cake", or "man kicking a ball"). Their method involves localizing the boxes containing the human and

the associated object, as well as identifying the action that is currently being performed, which ends up in the form of <human, verb, object>.

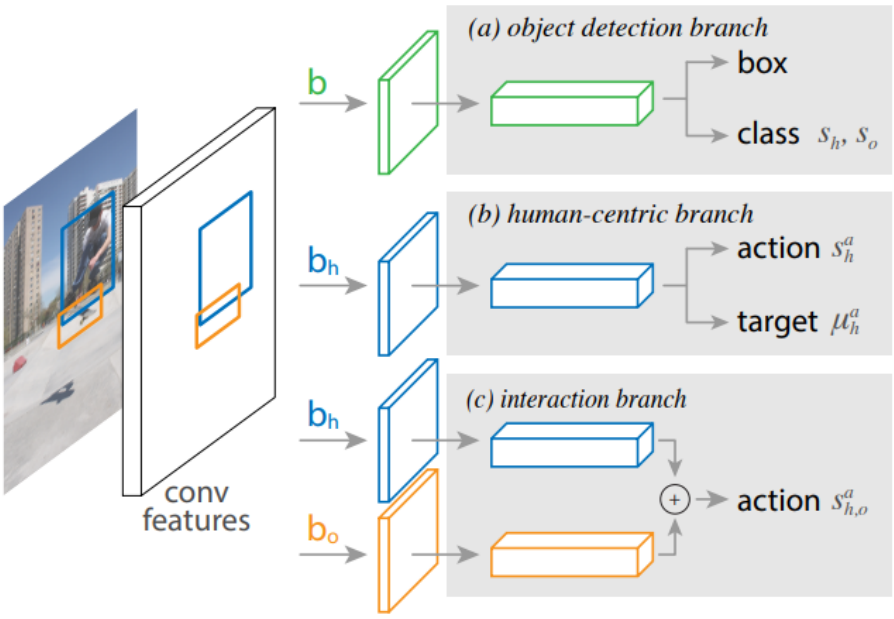

Figure 3.6: The model architecture used in [14], it consist of (a) an object-detection branch, a (b) human-centric branch, and an (c) interaction branch. The human features are shared between the (b) and (c) branch (blue boxes). Where the input, human-boxes, and object-boxes are denoted as b, bh, and bo respectively.

The object detection branch as seen in Figure [3.6] (a), which is exactly the same one used in [11] (Faster R-CNN) is responsible for detecting the objects in an image, followed by the branch (b), where its first role is to assign an action classification score to each human box, and action. Since a human can perform multiple actions at a time (e.g., sit and read) their output layer consists of multi-label action classification, while the second role of the branch is responsible for predicting the target object based on the person's appearance [14]. But, since the human-enteric branch focuses more on the person's appearance, it does not take into account the object's appearance [14], the interaction branch ,however scores the actions on both the humans and objects appearance.

3.3 Keras RetinaNet

RetinaNet was introduced by Facebook AI research (FAIR)[30], the one-stage detector that surpasses the two-stage detectors accuracy. RetinaNet is a composed network, that is composed of Feature Pyramid Network that has been built on top of the ResNet and its considered as the backbone of the RetinaNet to compute the feature maps for the entire image. Second we have the Subnetwork that uses the output of the backbone to perform the object classification. Thirdly, there is a second subnetwork that is responsible for the bound boxing regression by using the backbone's output [30].

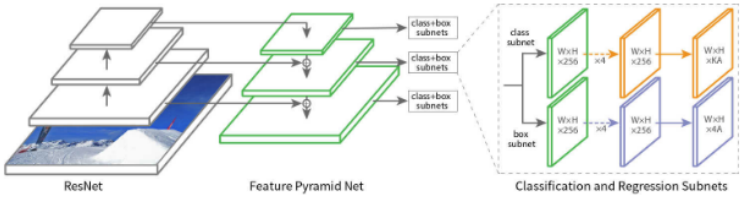

Figure 3.7: Visualization for the RetinaNet [30].

3.4 Summary

En conclusión, these papers showed us a lot of ways to detect objects, and how to detect the relationships between them. We will be going with the model described in [11] for object detection since we're not worried about real-time detection for now. For detecting relationships, we'll be using multiple models for different types of relations, we'll be using a semantic module to lower the number of possible relationships, a spatial module to detect spatial relationships, and lastly a visual module that is responsible for detecting every other relationship that the spatial module isn't detecting.

Chapter 4

Methodology

4.1 Exploring the Dataset

The dataset consists of 9,178,275 images. However, only 596,308 images have visual relationships labels, and of those, we will be using only 367,914 labels split among the train, validation, and test sets as such:

Table 4.1: Relationships and attributes.

	Train	Validation	Test	#Distinct relationship triplets	#Classes
Relationship triplets	348,560	4,951	14,403	1,384	288

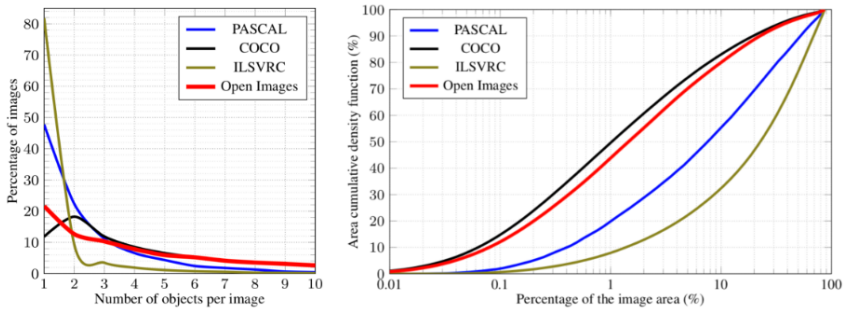

Figure 4.1: Number of objects per image (left) and object area (right) for Open Images.

It contains 9M images annotated with image-level labels (The image as a whole is annotated with meaning. Example: "Man on bicycle "), object bounding boxes, object segmentation masks, and visual relationships [6].

Figure 4.2: Sample from Open Images Dataset annotated with “Man on/top-of bicycle.

We’re only interested in the bounding boxes, and visual relationships so we won’t be using the whole dataset. It contains a total of 16M bounding boxes for 600 object classes on 1.9M images, making it the largest existing dataset with object location annotations. The object-bounding boxes have been manually drawn by a professional annotators to ensure accuracy and consisity. The images are very diverse and often contain complex scenes with several objects (8.3 per image on average). Open Images also offers visual relationship annotations, indicating pairs of objects in particular relations (e.g. "woman playing guitar", "milk on a table"). In total it has 329 relationship triplets with 391,073 samples. The dataset also have extra "Machine-Generated Labels" that split into training set (164,819,642 images), a validation set (681,179 images), test set (images 2,061,177) and 15,387 classes, this train images images have automatically generated by computer vision model, the validation and test sets, as well as part of the training set have human-verified image-level labels. Most verifications were done with in-house annotators at Google. This verification process practically eliminates false positives (but not false negatives: some labels might be missing from an image). The resulting labels are largely correct and we recommend to use these for training computer vision models, but we are not going to use them [6].

On average there are 8.4 boxed objects per image, 90% of the boxes were manually drawn by professional annotators at Google, using the efficient extreme clicking interface. This clicking interface works by clicking the image in the four physical points on the object: the top, bottom, left- and right-most points [6].

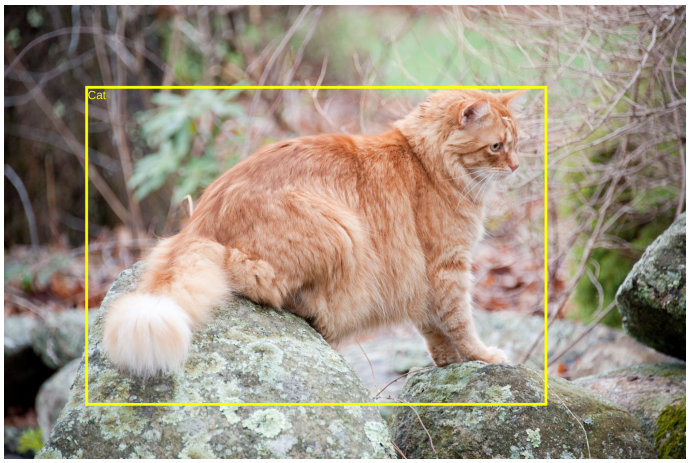

Figure 4.3: Image of cat labeled using Clicking-Interface.

After that, they drew a single box around groups of objects (e.g., a bed of flowers or a crowd of people) if they had more than 5 instances which were heavily occluding each other and were physically touching (we marked these boxes with the attribute "group-of").

For the training set, we annotated all images already containing bounding box annotations with visual relationships between objects and for some objects we annotated visual attributes (encoded as "is" relationship). A pair of objects connected by a relationship forms a triplet (e.g. "milk on a table"), Visual attributes are also represented as triplets, where an object in connected with an attribute using the relationship is (e.g. "table is wooden", "handbag is made of leather" or "bench is wooden"). They provide annotations exhaustively listing all positive triplets instances in that image. For example, for "woman playing guitar" in an image, we list all pairs of ("woman","guitar") that are in the relationship "playing" in that image. All other pairs of (woman,guitar) in that image are reliable negative examples for the "playing" relationship [6].

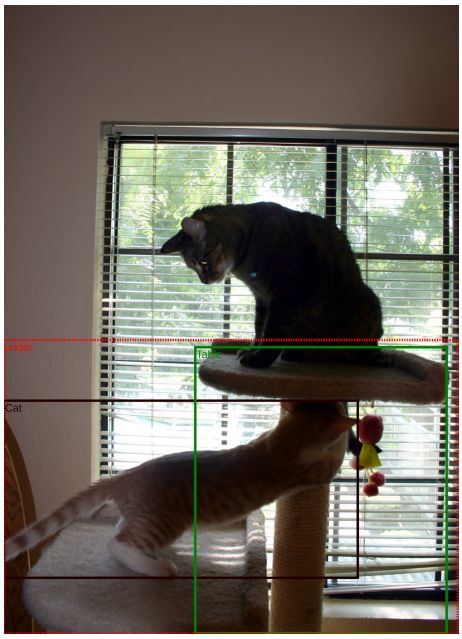

Figure 4.4: Cat under table" that has been annotated as relationship.

There are no more than 600 classes that ranges from foods to objects to animals and more.

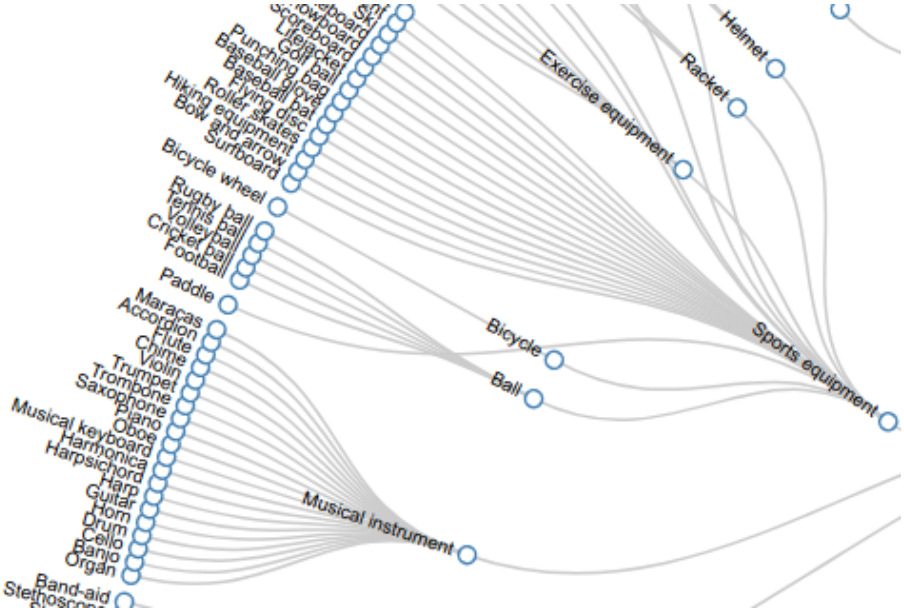

Figure 4.5: One side of the Hierarchy-view that describe the classes.

The visual relationships dataset consists of:

4.2 Approach

4.2.1 Object Detection

We used a pre-trained model to get the optimal input for our main goal which is relationship detection. The main approach of the model is splitting all classes to 5 levels depending on the child's level, which was done by using the official hierarchy Figure[4.6] that was provided with the dataset. [30]

The model was trained 5 times on each level, which resulted in 5 models in total. The classes were split in this manner due to the fact that the dataset has some images with only high-level classes which would confuse the model if it had to decide which object is which (Man -> Person).

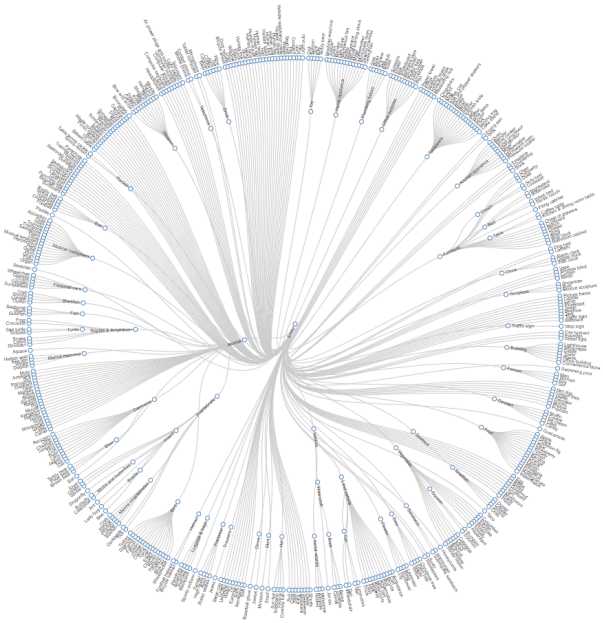

Figure 4.6: classes hierarchy [30].

However, he mentioned that having multiple levels doesn't increase the accuracy that much and only relying on the first level is good enough for having strong results, since they basically provide all the important classes.

The models he used were based on Keras RetinaNet which included a generator to train on this dataset, but he had to make several changes to it so that it suits his approach. The generator doesn't include augmentations for color so he added a random change in the intensity of the channels which resulted in better results.

4.2.2 Relationship Detection

In total the dataset includes ten kinds of relationships that include semantic, spatial, and visual relationships. Some examples of these relationships are, “wears”, “inside of”, “holds”, and “is”.

For this part of our approach we used GBM (Gradient Boosting Machine):

GBM Model: Following the previous model we will use a gradient boosting machine to decide what kind of relationship that we’re trying to predict. We will train this model on multiple things which includes:

- The objects bounding boxes and their area.
- The proposed relationship bounding box and its area.
- The euclidean distance between the two objects and their intersection over union.

The model is followed by another model that works as a protection layer that will be trained on the same features as the above model, but it will also include the relationship’s score from the previous model and the result of the prediction (1 or 0 depending on the output).

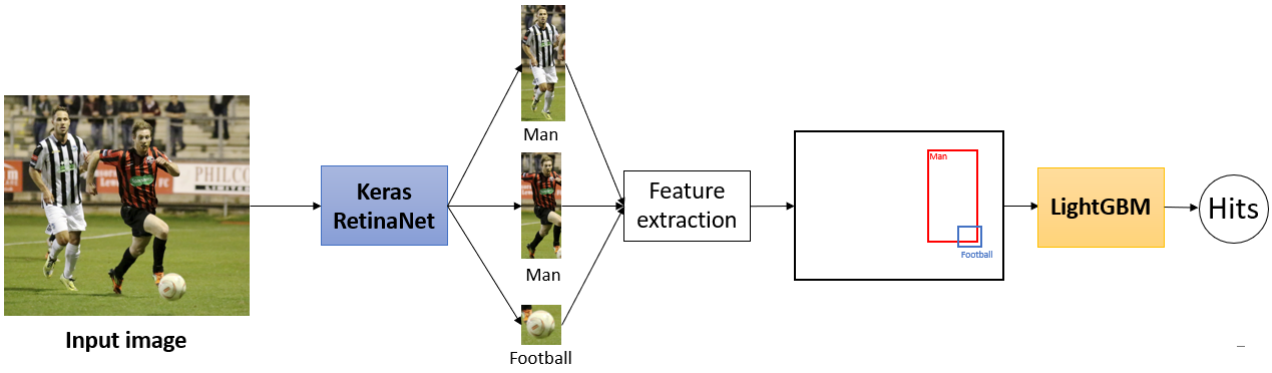

Figure 4.7: Our Approach Visualized.

Tools: We will be using Python for this project since it has very strong libraries that can help us in achieving our goals.

These libraries include:

- Pandas: Open source data analysis and manipulation tool that offers data structures and operations for manipulating numerical tables.
- LightGBM: A fast, distributed, high-performance gradient boosting framework based on decision tree algorithms, used for ranking, classification, and many other machine learning tasks.

4.3 Evaluation Metrics

To evaluate our work, we will be using a challenge dataset from a competition related to our dataset that consists of 99,999 images without any labels what so ever, which is evaluated using three evaluation metrics, with intersection-over-Union (IoU) threshold = 0.5:

- Mean Average Precision of relationships detection at IoU > threshold (mAPrel).
- Recall@N of relationships detection at IoU > threshold (Recall@Nrel).
- Mean Average Precision of phrase detection at IoU > threshold (mAPphrase).

mAPrel in relationships detection: For each relationship type (e.g. 'at', 'on') Average Precision (AP) a matching criteria must apply on the two object boxes and three class labels (two object labels and a relationship label). We consider a detected triplet to be a True Positive (TP) if and only if both object boxes have IoU > threshold with a previously undetected ground-truth annotation, and all three labels match their corresponding ground-truth labels. Any other detection is considered a False Positive (FP) in the two cases (1) both class labels of the objects are annotated in that image (regardless of positive or negative); or (2) one or both labels are annotated as negative. Finally, if either of the labels is unannotated, the detection is not evaluated (ignored). mAPrel is computed as the average of per-relationship APs [24].

Recall@Nrel in relationships detection: The triplet detections are sorted by score and then the top N predictions are evaluated as TP, FP or ignored (see above). A recall point is scored if there is at least one True Positive is found among these top N detections [24].

mAPphrase in phrase detection: Each relationship detection triplet is transformed so that a single enclosing bounding box is formed from the two object detections. This bounding box has three labels attached (two object labels and one relationship label). The enclosing box is considered to be a TP if IoU > threshold with a previously undetected ground-truth annotation and all three labels match their corresponding ground-truth labels. The AP for each relationship type is computed according to the PASCAL VOC 2010 definition. mAPphrase is computed as the average of per-relationship APs [24].

Chapter 5

Experimental Design

5.1 Feature Engineering

During the making of this report, we were very focused on how to represent the raw data that we had to work with into this features that could help our predictive model understand the problem fully and make use of some of the unseen data.

5.1.1 Input Features

When it was time to take a look at the data we figured that we had a lot of information. However, we needed to feature engineer it a little more to bring out everything that we need. The input consisted of the two objects labels, their corresponding bounding boxes, and the relationship label that connects them.

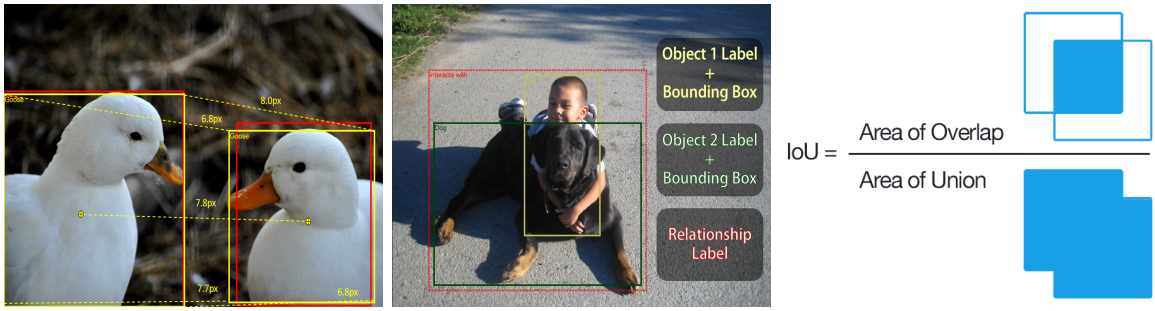

Figure 5.1: A visualization of the data that we had to work with.

During feature engineering, we experimented a lot and this is what we found that most affect our goal:

- **Relationship Bounding Box:** We used the objects bounding boxes to create a larger bounding box that represents the whole relationship.

- **Bounding Boxes Area:** We calculated each bounding box's area which resulted in 3 different areas to work with.
- **Distance Between the Two Objects:** For the two objects we calculated the euclidean distance between each corner and their center.

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

- **Intersection over Union (IoU):** We also added the IoU of the two objects as an input to try and represent the overlapping of the objects.

$$IoU = (B_1, B_2) = \frac{B_1 \cap B_2}{B_1 \cup B_2}$$

- **Negative Sampling:** We generated negative samples by going through all images that we were using and connected every possible object pair that wasn't included in the relationship dataset.

5.1.2 Class Imbalance

Due to the multiclass nature of our problem, we faced many issues that can be easily represented in Figure[5.2]. As we can clearly see our target classes are not nearly balanced in a way that is usable for us.

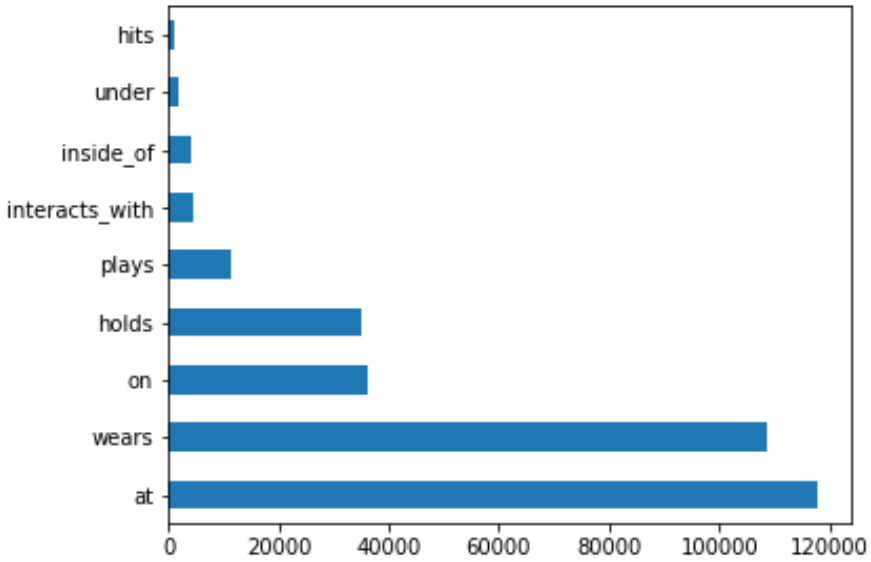

Figure 5.2: Visualization of the class imbalance that we have.

This would eventually be a problem when it came time for training, so we came up with two solutions for this problem to help us in restoring a bit of balance between the classes without losing much data:

1. **Undersampling:** By adjusting the class distribution of the dataset, we balanced the ratio between the classes to a much more reliable sample.
2. **Combining Similar Relationships:** We quickly noticed that we have a few relationships with similar meanings to each other (i.e "under" is similar to "on" and "hits" is similar to "interacts with"), so we merged the smaller count relationships with the largest.

In the end, we combine both our input features and the above solutions to create our final dataset which we used to train our model.

5.2 Implementation Issues

While working on this project, we faced many problems that we needed to solve in a short matter of time and being inexperienced in the field wasn't helping us so we had to research a lot on these topics and make tough decisions on the way.

5.2.1 Processing Open Images Dataset

We have faced many problems with the dataset due to its large size, the dataset consists of 1.8M images with the size of 561GB. We couldn't process these images with our limited resources and had to rely on cloud services to reach our goal.

The dataset wasn't very clean with it having missing labels on some images and it having balance issues on both the object detection part and relationship detection part.

Luckily, GCP (Google Cloud Platform) has a free trial with $300 credits, that we took advantage of to train our initial model for object detection until we decided to change our approach to include a pre-trained model to detect the objects in an image.

We made this change because after researching the topic we found out that a tree-based approach here yields better results than neural networks.

5.2.2 Processing the Challenge Dataset

When it was time to process the 99,999 images for the challenge, it needed to take 300 hours to finish. So we distributed the images among 5 machines to cut down on the wait time and get results faster with each submission to the competition.

5.2.3 Library

At first, we started working with PyTorch because it was user friendly and great with computer vision applications, but our approach changed during the making of this report

from using neural networks to using decision trees which are implemented very efficiently in LightGBM.

The learning curve between the two libraries isn't that big since LightGBM is very easy to use and provided a relatable API to ease us into starting our training in no time.

5.2.4 Hyper Parameter Tuning

When we first started training our model, our results were very bad and unusable. So, we had to tune our parameters in a way that would benefit us the most.

We experimented a lot during this part and ran a grid search on the parameters that we believed affected the model the best. The results, in the end, were very satisfying and easily beat our expectations.

Chapter 6

Results and Discussion

6.1 Results

For this project, we had two different ways to measure results. The first being is we ran our models on a small set of the data. By doing this we don't need to worry about bounding boxes not being accurate or misleading which makes evaluation very easy.

The other evaluation method that we used was to submit our predictions of the challenge set and getting their score that is evaluated using the evaluation metrics discussed from before.

Table 6.1: Classification report

Relationship	Precision	Recall	F1-Score
on	0.88	0.98	0.93
plays	0.87	0.29	0.44
inside_of	0.98	0.87	0.92
interacts_with	0.87	0.77	0.82
at	0.99	1.00	1.00
holds	0.77	0.83	0.80
wears	0.98	1.00	0.99
accuracy	–	–	0.97
macro avg	0.91	0.82	0.84
weighted avg	0.94	0.94	0.94

6.1.1 Classification Report

Evaluating the models this way we only need to check the model's output since we're pretty sure our input is correct and doesn't have any problems.

We calculated precision and recall for each class separately to get a better look at how each class is being treated and predicted.

$$Precision = \frac{(true\ positive)}{(true\ positive) + (false\ positive)}$$

$$Recall = \frac{(true\ positive)}{(true\ positive) + (false\ negative)}$$

We even calculated the f1-score to measure our test's accuracy which takes precision and recall into account which results in combining them and giving us a score that depicts how well our model is performing.

$$F_1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

As we can see in Table[6.1] most classes have a pretty high-score except for "plays", "interacts-with", and "holds", which could be because we don't have many samples for them.

You may even notice that we're missing some of our targets, and that's because we merged some of the targets together.

Nevertheless, even though our models need a little extra work to predict lesser targets, our accuracy overall is pretty high and we're pretty satisfied.

6.1.2 Challenge Set Results

For the other part of our results, we were left with a bad score and we have a few theories as to why that is, even though during training our results were pretty high.

Table 6.2: Submission results.

	Private Score	Public Score
First Submission	0.03423	0.03629
Last Submission	0.03675	0.03812

As you can see our score is very low, even though it increased over all of the submissions that we made it was still considered low. Since for the challenge set we used a pre-trained model for object detection, we strongly believe that it works as the core part for this challenge because the relationship predictions heavily rely on its output.

Also, our model was forced to work with new objects that it hadn't faced before which messes up the weights when making a decision.

The scores don't include the usage of the low count targets which potentially hurts our score, but we wanted to focus more on the relationships that we were good at predicting.

6.2 Discussion

We will split this into two different parts, one for discussing our results which were from our classification report, and the other is for the competition's results.

6.2.1 Model Tuning and Feature Engineering

We think that our model could perform better if we tuned our parameters a little better, by applying methods such as random search or grid search in new ways that could potentially provide better results.

Also, we think that we handled some parts of our model, specifically the parts with under-sampling and combining targets very poorly, and we could've gotten better results if we tried different methods for balancing.

Perhaps having multiple models that could help us with each target followed by an ensemble of their predictions, would probably yield better results overall, and would solve our imbalance issues because we would be training each target with its positive samples and generate negative samples from the images that the target appears in.

6.2.2 Object Detection Model and Categorical Features

For the competition, we had to rely on a pre-trained model for object detection and we had no control over it. This resulted in us having a very bad score because we believe that the core part of relationship detection is detecting the objects in the first place.

We could solve this by having our own object detection model which we need to train until we reach a point where we are satisfied with it. Which hopefully would increase our overall score with the challenge set.

Another problem we faced with the pre-trained model is that we had to use objects that our model had never seen before, which in our case would be handled by always going to the right of the nodes that rely on the object labels in the decision tree that is implemented in LightGBM.

This could be solved by better encoding our categories in a way that could easily work with new objects it faces, by probably joining categories into a bigger category that could be used to represent many different categories.

Chapter 7

Conclusión

Object relationship detection in image processing is not an easy task to do, there are many steps that go into achieving this goal. There have been many approaches to achieve this, but most of them are limited in the number of relationships since there are so many kinds of relationships to consider, and trying to train for all of them is one of the most difficult tasks to do in computer vision. In this paper, we discuss some of the modern approaches and propose a solution of our own, that we apply on the Open Images open-source image dataset.

Bibliography

[1] D. MICHIE, "Memo functions and machine learning," p. 1, 1968.

[2] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, http://www.deeplearningbook.org.

[3] M. Taylor, *Neural Network: visual introduction for beginners*. Independently, 2017.

[4] "Deep learning," https://www.investopedia.com/terms/d/deep-learning.asp.

[5] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," 2016. [Online]. Available: https://doi.org/10.1109/CVPR.2016.91

[6] "Open images dataset v5 + extensions," https://storage.googleapis.com/openimages/web/index.html.

[7] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," 2012.

[8] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," 2014.

[9] R. Girshick, "Fast r-cnn," 2015.

[10] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," 2016.

[11] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," 2018.

[12] C. Lu, R. Krishna, M. Bernstein, and L. Fei-Fei, "Visual relationship detection with language priors," 2016.

[13] G. Gkioxari, R. Girshick, P. Dollar, and K. He, "Detecting and recognizing human-object interactions," 2018.

[14] "Gradient boosting," https://bradleyboehmke.github.io/HOML/gbm.html.

[15] "Deep learning," https://www.nature.com/articles/nature14539.

[16] A. Rosebrock, *Deep Learning for Computer Vision with Python*, 2017.

[17] L. Weng, "Object detection for dummies part 1: Gradient vector, hog, and ss," *lilianweng.github.io/lil-log*, 2017. [Online]. Available: http://lilianweng.github.io/lil-log/2017/10/29/object-recognition-for-dummies-part-1.html

[18] "Histogram of oriented gradients," https://www.learnopencv.com/histogram-of-oriented-gradients/.

[19] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation," 2004.

[20] L. Tindall, C. Luong, and A. Saad, "Plankton classification using vgg16 network," 2017.

[21] "A simple explanation of the softmax function," https://victorzhou.com/blog/softmax/.

[22] A. Kuznetsova, H. Rom, N. Alldrin, J. Uijlings, I. Krasin, J. Pont-Tuset, S. Kamali, S. Popov, M. Malloci, T. Duerig, and V. Ferrari, "The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale," *arXiv:1811.00982*, 2018.

[23] "Open images challenge object detection evaluation," https://storage.googleapis.com/openimages/web/evaluation.html#visual_relationships_eval.

[24] "The pascal visual object classes homepage," http://host.robots.ox.ac.uk/pascal/VOC/.

[25] "Cocodataset/," http://cocodataset.org/.

[26] "Large scale visual recognition challenge (ilsvrc)," http://www.image-net.org/challenges/LSVRC/.

[27] H. Rowley and S. Baluja, "Neural network-based face detection, pattern analysis and machine intelligence, ieee transactions on 20(1)," 1998.

[28] P. Sermanet, K. Kavukcuoglu, S. Chintala, and Y. LeCun, "Pedestrian detection with unsupervised multi-stage feature learning. in: Proceedings of the ieee conference on computer vision and pattern recognition," 2013.

[29] "15th place solution," https://www.kaggle.com/c/google-ai-open-images-object-detection-track/discussion/64633.

[30] "Retinanet explained and demystified," https://blog.zenggyu.com/en/post/2018-12-05/retinanet-explained-and-demystified/.