Finding Waldo: An Investigation into the Applications of Machine Learning in Object Detection

Jonathan Camarillo, Maria Contreras, Mayleen Cortez, Andrew Martos • Dr. Alona Kryshchenko

Channel Islands
CALIFORNIA STATE UNIVERSITY

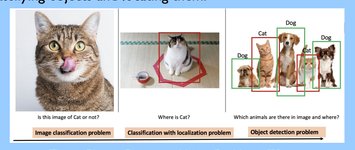
Background and Motivation

Project and Motivation

- **What?** Teach a computer to find the Waldo character from the children's series *Where's Waldo*[1]
- **How?** Using the Single Shot multibox Detector algorithm to train a machine learning model
- **Why?** To motivate further investigations into machine learning for object detection
- **Why Waldo?** Since Waldo is an iconic character, this project may spark student interest in data science, computer science and mathematical research

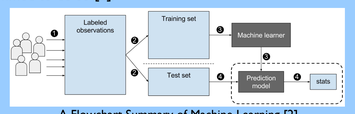
Background

• Computer Vision is a field of research that deals with training computers to do the high-level visual processes of humans, such as accurately classifying objects and locating them.



Three Classic Computer Vision Problems [2]

• Machine learning algorithms use data, experience and interaction to teach a computer how to learn and improve without being directly programmed to do so [2].



A Flowchart Summary of Machine Learning [2]

• According to Google, cloud computing is "the practice of using a network of remote servers hosted on the Internet to store, manage and process data, rather than a local server on a personal computer."
  - Cloud computing is preferred for training machine learning models because of the large datasets involved
• Of the various cloud-computing services for us to use for object detection (Google Cloud Machine Learning, IBM Watson, Azure Machine Learning), we used Amazon Sagemaker via Amazon Web Services (AWS).

[1] *Where's Waldo* was created by Martin Handford. The rights to the Waldo character are held by DreamWorks.

Goals

- To create a machine learning model that uses the Single Shot multibox Detector algorithm to find the Waldo character from the children's book series *Where's Waldo* with at least 90% accuracy
- To train and implement our machine learning model using AWS as our platform
- To utilize or incorporate Amazon web Services' DeepLens in our project
- To explore the applications of our work
- To increase student interest in computer science, data science and mathematics research

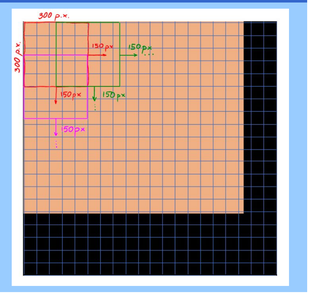
Data Collection

	A	B	C	D	E	F	G
1	Filename	x_1	y_1	BB Width	BB Height	Image Width	Image Height
2	w1.jpg	47	19	54	67	300	168
3	w2.jpg	1540	367	43	46	1843	1309
4	w3.jpg	1388	465	37	38	72	72
5	w4.jpg	372	381	17	22	800	600

- To train a machine learning model that works in computer vision, we need images to train and validate the model with.
- We split the Waldo images into two sets.
- For each image, we must know the "answer," i.e. where Waldo is.
- To do this, we label each image with a bounding box (BB).
- An example of how this looks is to the left. The top photo comes from [4] and the bottom photo comes from [5].
- Above, you see an example of the information, or labels, we need for each image, e.g. BB coordinates and image dimensions.
- For AWS SageMaker, this information must be in a text file formatted in special way. Altogether, it is called the JSON file.
- During the training phase, the model utilizes images from the training set and the labels associated with them.
- Using this information, it estimates a function to predict when something is a "Waldo" or "Not Waldo."
- Then, it uses the images from the validation set to make predictions without first looking at the answers.
- Finally, the model compares its results to the answers (labels) to determine if the model was successful or if we must retrain.

Data Augmentation

- We are able to increase the number of Waldo images that we have using certain techniques.
- For example, we can horizontally flip all the images. This means we reflect the image over the y-axis.
- In digital image processing, images are treated like a coordinate plane with the origin at the top left corner.
- Another way to augment the data is to "chop up" the original image in a special way, shown in the image on the left.



Single Shot multibox Detector

- SSD is an object detector that uses the VGG16 network as its base. Object localization and classification are done in a single forward pass. SSD uses a technique derived from MultiBox for bounding box regression.
- During training, SSD takes in images that have associated ground truth boxes for each object in the images.
- SSD associates a set of default bounding boxes to every feature map cell for multiple feature maps.
- These default boxes are carefully and manually chosen.

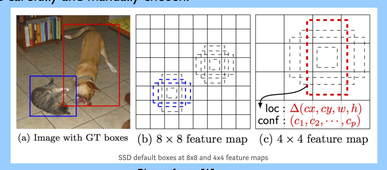


Photo from [1]

- Shape offsets and confidences are predicted for each object in the default boxes.
- The model's overall loss is computed by taking a weighted sum of the confidence loss and the localization loss. This can be interpreted as how far away the feature map is from the actual ground truth box.

$$L(x, c, l, g) = \frac{1}{N}(L_{conf}(x, c) + \alpha L_{loc}(x, l, g))$$

- Matching Strategy: When SSD is being trained, the default boxes that correspond to the ground truth boxes need to be determined. This matching strategy is done by matching each ground truth box to the default boxes with the best intersection to union ratio (IoU also known as jaccard overlap). Finding the overlap is important because this tells us how close we are to the ground truth box. Default boxes are matched to any ground truth boxes with an IoU higher than a threshold of 0.5.

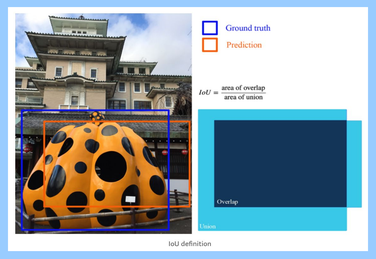


Photo from [6]

- After the matching strategy, most of the default boxes are negatives. This gives us a significant imbalance between positives and negatives. Hard negative mining is done to sort through all of the default boxes with the highest confidence loss and pick the top ones so that the ratio between the negatives and positives is at most 3:1.

Architecture for SSD:

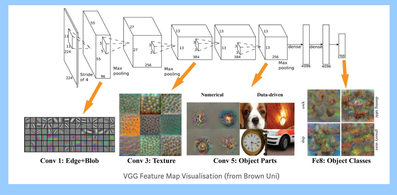


Photo from [3]

Results

- We trained twelve models each with different hyperparameters
  - Batch size, epochs, and learning rate
- Used cross-validation technique to evaluate models
  - Training set contained 120 images
  - Validation set contained 40 images
- Selected two with the highest mean average precision (mAP) score
- Trained each model three times without changing their hyperparameters and selected the one with the best average mAP score
- The model was trained five more times
- Best Model Hyperparameter Settings
  - Batch Size = 10; Epochs =120; Learning Rate = 0.00082
- Best Model mAP scores from eight trials
  - 0.929; 0.918; 0.900; 0.895; 0.884; 0.858; 0.830; 0.822
- Average of the eight mAP scores = 0.880
- Tested the model using nine unique images
- Difficulty was determined using Waldo's body size, head size, resemblance to other image entities, and general variance of color and patterns
- The detection probability and bounding box accompanies each image

EASY **MEDIUM** **HARD**

Acknowledgements

- We would like to thank Dr. Alona Kryshchenko for her continued guidance and support with this project.
- We would also like to thank the CSUCI Mathematics department for printing our poster and providing us funds to use AWS.

Citations

[1] Berg, Alexander C., et al. *SSD: Single Shot MultiBox Detector*. 2016, arxiv.org/pdf/1512.02325.pdf.
[2] EpochFail., https://creativecommons.org/licenses/by-sa/4.0/legalcode
[3] Grover, Prince. "Evolution of Object Detection and Localization Algorithms." *Towards Data Science*, Towards Data Science, 15 Feb. 2018, towardsdatascience.com/evolution-of-object-detection-and-localization-algorithms-e241021d8bad.
[4] Handford, Martin. *Where's Waldo?* Candlewick Press, Massachusetts. 2007. Print.
[5] Handford, Martin. *Where's Waldo? In Hollywood*. Candlewick Press, Massachusetts. 2007. Print.
[6] Hui, Jonathan. *mAP (mean Average Precision for Object Detection)*. 2018. Online. https://medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121a31173