EG UK Computer Graphics & Visual Computing (2021)
K. Xu and M. Turner (Editors)

Where's Wally? A machine learning approach

T. Barthelmes and F. P. Vidal[†]

School of Computer Science & Electronic Engineering, Bangor University, UK

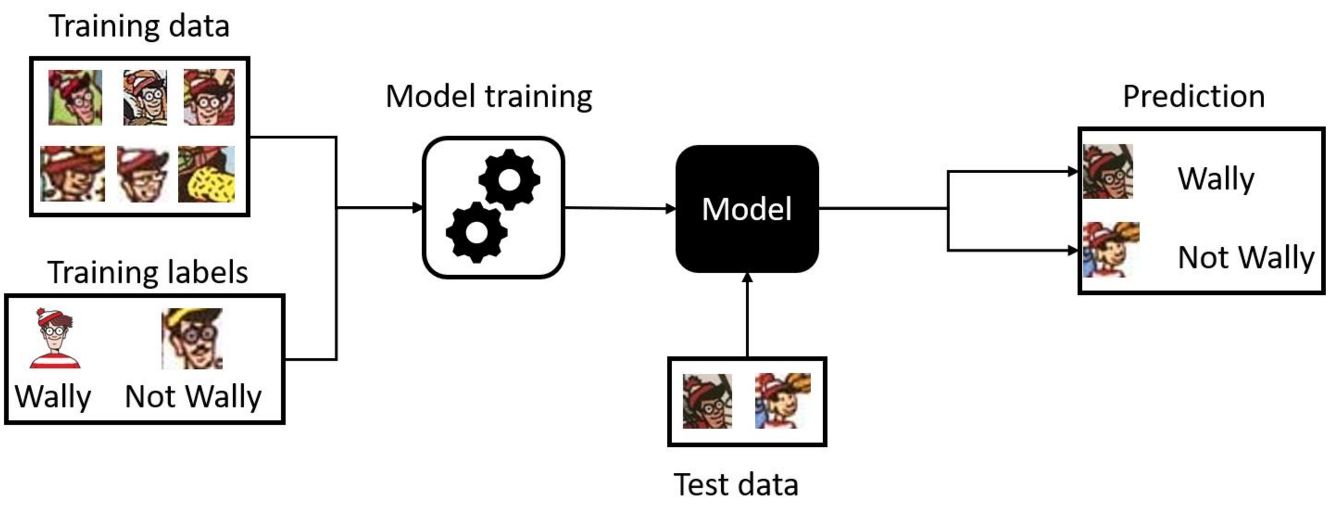

Figure 1: *Example of supervised learning where the model learns from labelled data to be able to classify test data.*

Abstract

Object detection has been implemented in all sorts of real-life scenarios such as facial recognition, traffic monitoring and medical imaging but the research that has gone into object detection in drawings and cartoons is not nearly as extensive. The Where's Wally *puzzle books give a good opportunity to implement some of these real-life methods into the fictional world. The Wally detection framework proposed is composed of two stages: i) a Haar-cascade classifier based on the Viola-Jones framework, which detects possible candidates from a scenario from the* Where's Wally *books, and ii) a lightweight convolutional neural network (CNN) that re-labels the objects detected by the cascade classifier. The cascade classifier was trained on 85 positive images and 172 negative images. It was then applied to 12 test images, which produced over 400 false positives. To increase the accuracy of the models, hard negative mining was implemented. The framework achieved a recall score of 84.61% and an F1 score of 78.54%. Improvements could be made to the training data or the CNN to further increase these scores.*

CCS Concepts

• ***Computing methodologies*** → ***Machine learning;*** • ***Applied computing*** → *Media arts;*

1. Introduction

Where's Wally is a collection of puzzle books where the reader has to find Wally, shown in Figure 2, hidden in a drawn image containing lots of characters doing silly things. Every page has a different scenario where it gets progressively harder to find Wally. Wally may be partially obscured by a person or object. Sometimes it is possible to see his whole body, other times only his head can be seen. Wally has a distinctive style of red and white along with a bobble hat but sometimes the illustrations can contain a ruse of other red-and-white stripped objects to challenge the reader.

Our research questions can be summarised as follows: *Are face detection and face recognition of cartoon characters possible*

[†] Secretary UK Chapter of the Eurographics Association

DOI: 10.2312/cgvc.20211313

Figure 2: *Portrait of Wally from the Where's Wally book [Han87].*

with computer vision and machine learning techniques that were initially developed for real-life scenarios? To the best of the authors' knowledge, to this day, not much work has gone into detecting objects in drawings or cartoons and no formal research have been published of any Wally detection system.

Section 2 describes related work, focusing on facial recognition. It starts with traditional techniques and ends with today's most popular approach, namely the use of convolutional neural networks (CNNs). In Section 3, we present the data repository that was used to provide the training and testing data. It is followed by a description of our method, which makes use of a combination of a more traditional approach with a modern CNN. The paper ends with a conclusion in Section 5.

2. Related Work

Finding Wally is in essence a facial recognition problem. Luckily, facial recognition is a large sub topic of computer vision (CV) and many approaches have been developed. The two main object recognition paradigms that have emerged are machine learning (ML) and deep learning (DL) [KKSK20]. The ML approach trains a classifier on predetermined features [Kot07] such as Haar-like [VJ01] or histogram of orientated gradients (HOG) features [DT05] and a DL approach learns for itself the best features to classify an object with [AMAZ17].

2.1. Haar Cascade Classifier

One of the most famous facial recognition frameworks devised is that of Paul Viola and Michel Jones which trains a cascade classifier on Haar-like features [VJ01]. This idea was motivated from Haar wavelets, which are a series of rectangle-shaped functions. Haar-like features describe the difference of pixel intensities in rectangular regions located next to each other. These regions can describe images in different ways as shown in Figure 3.

To calculate the difference between the adjacent rectangles for a feature in a time complexity of $\mathcal{O}(1)$ an integral image is used, which is defined as the sum of all the pixels of the element above and to the left of the current element. A rectangular sub-region can be calculated using only 4 look-ups of the integral image as shown in Equation 1. The letters represent the coordinate of the corners of a rectangle in an image. However, Haar-like features have more than four corners so require between six to nine look-ups, feature dependent.

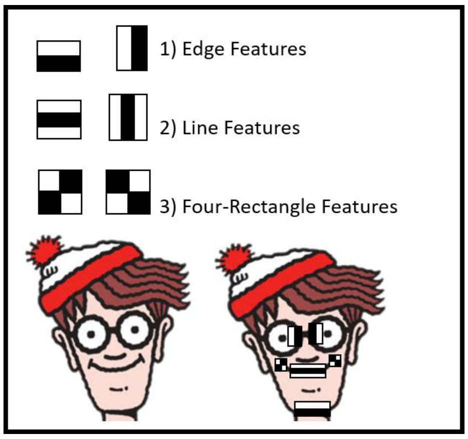

Figure 3: *An example of how 1) Two-rectangle features, 2) Three-rectangle features and 3) Four-rectangle features are used to identify an object.*

$$\sum_{\substack{x_0 < x \leqslant x_1 \\ y_0 < y \leqslant y_1}} i(x,y) = I(D) + I(A) + I(B) - I(C) \tag{1}$$

An ideal feature will have a difference of one. The closer to one the more likely that the feature is a Haar-feature. Thus, a threshold can be calculated to determine what should be classified as a Haar-like feature. To find the best features, Viola-Jones implemented the Adaboost algorithm. This applies every feature to all the training images. It then selects the features which best separate the positive images from the negative images. However, classification of images would take a lot of time, if each of the selected features were to be applied to every image. As such, the features are grouped into weak classifiers, which by themselves are not enough to label an image as the object, but are enough to label an image as definitely not the object. When these classifiers are grouped into a cascade they become a strong classifier which can positively label an object in an image.

2.2. Deep Learning

Deep learning models such as CNNs have become the forefront in object recognition due to there ability to learn an objects features and label them correctly to a high accuracy [AMAZ17]. However, a lot of data and computational power are required to train an accurate network.

The first layer is the convolutional layer which applies a filter via a kernel to the image which allows it to extract a feature map. Different filters extract different features, thus the more convolutional layers a neural network has the more features it can extract. Before the next layer the output of the convolution layer is put through the Rectified Linear Unit (ReLU) activation function. The function sets all negative values to zero which allows for a further reduction in processing without generalising the data. After every convolutional layer comes a pooling layer which down sizes the sample of the feature map to reduce the amount of processing required by either selecting the maximum value or the average value of a pre-defined grid in the feature map. The last layer of processing in CNN is the fully connected layer. This looks at which features correlate to which objects by multiplying the weights with the previous layer in order to get the correct percentages for the different classes. The class with the highest percentage is what the object will get labelled as. In order to train a CNN, the filters and weights are created first based on Gaussian distribution. As a result, initially a lot of images are miss-classified. However, much like humans, CNNs learn from their mistakes. In the case of neural networks, this happens via a process called backpropagation. This goes through the network from end to start biasing the filters and weights to correctly label the images.

3. Data Repository

A data set of images of Wally and of Not Wally images had to be created for use of training the models. The author of the books made it a challenge to find Wally, he is never exactly the same in every scene. As a result, lots of images of Wally from different sizes, angles and occlusions are required to train a classifier which can consistently detect Wally. The author of the books also increased the challenge of finding Wally by adding lots of similar looking characters. This means that the negative data set will have to contain many of these Wally look-alike characters along with random characters from the *Where's Wally* books.

Most of the images come from a GitHub repository [Con19], however, due to the lack of images of Wally, we went through the first *Where's Wally* book [Han87] and took screenshots of Wally from different pages. The positive data set contains 84 different images of Wally and the negative data set contains 184 images of objects which are not Wally.

4. Methods

Our approach makes use of two classifiers staked on top of each other as shown by Figure 4. The first is a cascade classifier trained on Haar-like features to select a sub-sample of potential Wally objects from the scene. The second is a CNN which picks out Wally from the selection provided by the cascade classifier. Whilst this approach is not new, we propose here to apply it onto cartoon characters [LLS*15, DX17]. The advantage of this strategy over an end-to-end CNN is to quickly reject the background, hence improving the computational performance. As this is only a prototype solution a Python machine environment was used so that any solutions could be quickly created.

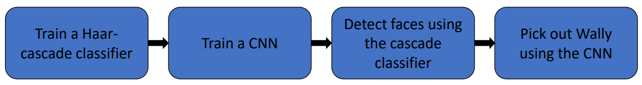
Train a Haar-cascade classifier
Train a CNN
Detect faces using the cascade classifier
Pick out Wally using the CNN

Figure 4: *The pipeline showing the steps that were needed to be taken to create and implement a Wally detection system.*

4.1. Haar-cascade classifier

The Haar-cascade classifier is trained using OpenCV's command line program. It calculates the relevant Haar-like features in the images of the training data set to create a cascade classifier using AdaBoosting. It uses these to create a cascade model which can be used to detect faces. After training has occurred the false alarm rate and acceptance ratio is given to get an indication of how over-trained the classifier might be. Overfitting occurs when the acceptance ratio is less than 10^{-5}. The alarm rate corresponds to the amount of false positives being passed at every stage of the classifier. The classifier that was created had an alarm rate and acceptance ratio of 0.000385 and 0.000564 respectively. However, the best indication of how the classifier performed was by applying it to the 12 test images. The classifier found Wally in every image and also found a total of 442 false positives in all the images. Figure 5 shows some of the detections made by the Haar cascade classifier in the first test image. The classifier can not differentiate Wally objects from Wally-like objects which is why a further model had to be added to select Wally from the detected objects.

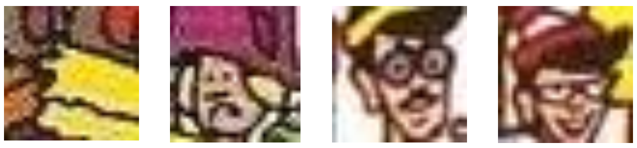

Figure 5: *A selection of false positives made by the framework.*

As the Haar-cascade classifier is not enough to exclusively select Wally from the images a CNN was created and trained to select Wally from all the false positives. We evaluated two approaches. The first one makes use of two popular convolutional neural network architectures. The second is a custom CNN architecture.

4.2. Off-the-shelf CNN architectures

We use two popular off-the-shelf CNN architectures: Residual Neural Network (ResNet) and MobilNet, from TensorFlow. These models are both variations of deep CNNs with pre-defined architectures and weights. Residual Neural Network (ResNet) is a deep CNNs which reduces the issue of vanishing gradient by implementing skip connections between layers which maintain a local gradient of one [HZRS16]. MobileNet was made for mobile applications and provides a lightweight CNN model by reducing the number of parameters involved in the calculations [SWA19].

The same data set was used to train both the off-the-shelf models and the custom model which included 442 negative images and 72 positive images. We then split the data-set into a training and testing sub-set with 80% of the data being used for training and 20% of the data being used for testing. To increase the accuracy of the models, hard negative mining was implemented to ensure the same false positives were not detected [SWH18].

Only Wally objects with a probability over 90% were considered to be Wally by the framework. Before any of the training images could be used for training and testing they were all resized to the average height of all the training images, then all the pixel values of the images were normalised to a range between zero and one.

Table 1 shows the accuracy, precision, recall and F1 measures for both models. Accuracy is the ratio of correctly labelled objects to the total number of detected objects. However, this value can be misleading if the classification of a small small class is vital as the score can still be very high. Recall and precision offer a better insight into classifier performance for this case. Recall refers to the percentage of true positives from all the labelled objects (i.e. how many of the detected objects are correctly labelled Wally) and precision refers to the percentage of true positives from all the positively labelled results (i.e how many of the objects labelled Wally are actually Wally). The higher the recall the lower the precision. The F1 score is a harmonic mean of both recall and precision and gives an overall indication of how balanced the precision and recall measurements are. The precision and accuracy scores for MobileNet are close to or almost 100%. This shows that this model labelled all the true negative objects correctly but failed to label any of the true positive objects as the recall score is zero. This shows the class is unbalanced and as such the F1 score has to be taken into account. The F1 scores for both ResNet50 and MobileNet are 0% which is why an alternative custom model to detect Wally was created.

4.3. Custom CNN model

The CNN that was implemented is made up of three convolutional and pooling layers with ReLU activation followed by a dropout, flatten, and fully connected layer. All convolutional layers have a kernel of size 3×3. The first convolutional layer extracts 32 feature maps from an image of size 44×44. The other two convolutional layers extract 64 feature maps. All three pooling layers calculate the maximum value of every 2×2 window in the image to reduce the size of the feature map without generalising the data. The dropout layer sets input units to zero randomly at a rate of 0.2 during training to prevent overfitting. The fully connected layer contains 64 units which all use ReLU activation. The model was compiled using the Adam optimiser and calculates the difference between the predicted label and the true label using sparse categorical cross-entropy. The Adam optimiser quickly converges to the lowest loss for a function by updating the learning rate based on the momentum of every parameter in the function at every epoch during training [KB14]. The model was then trained for 12 epochs and subsequently added to the framework.

When the framework was applied to the 12 test scenes it detected Wally in ten of those images. In every test scene no false positives were detected. Table 1 shows how well the model performed. It highlights that the model prioritises recall over precision which means that Wally is more likely to be labelled as Wally, than it is as not Wally, but also that more non-Wally objects will be labelled as Wally. Another reason why the discrepancy between the recall and precision is not of a concern is because the F1 score is relatively high. When the framework was implemented on the first test image, it only found Wally and no false positives as shown in Figure 6.

Table 1: *Results of the different CNN models when applied to the test images*

Model	TP	TN	FP	FN	Accuracy	Precision	Recall	F1
ResNet50	0	0	442	13	0%	0%	0%	0%
MobileNet	0	442	0	13	97.14%	100%	0%	0%
Our custom method	11	437	0	2	98.46%	73.34%	84.61%	78.54%

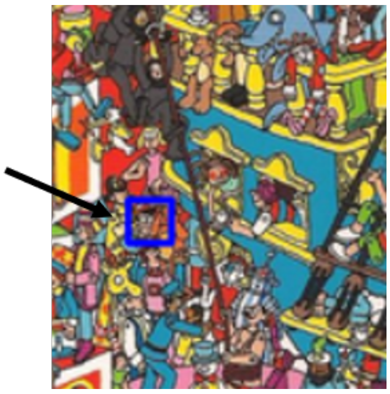

Figure 6: *All the detections made once the CNN has been applied to the framework. The arrow is pointing towards Wally in the image.*

5. Conclusions

We have shown that it is possible to deploy the Haar-cascade classifier to detect faces of cartoon characters, and CNNs to recognise Wally. Overall, the framework has successfully detected Wally in most of the test scenes to a high degree without also detecting objects similar to Wally. Computer vision and machine learning techniques initially developed for real-life scenarios can therefore be applied on drawings and cartoons.

The training dataset used here is relatively small, which may limit the performance of the proposed solution. Future work will include the evaluation of our framework with a dedicated benchmark dataset of cartoon faces [ZZR*20], and compare it in term of classification results and computational performance with an end-to-end CNN.

Acknowledgements

Part of this work has been submitted in partial satisfaction of the requirements for the Degree of Bachelor of Science (BSc) in Computer Science at Bangor University, UK, for which F. P. Vidal's support was easier to find than Wally much to my sincere appreciation (TB).

Acronyms

CNN	convolutional neural network.
CV	computer vision.
DL	deep learning.
FN	false negative.
FP	false positive.
HOG	histogram of orientated gradients.
ML	machine learning.
ReLU	Rectified Linear Unit.
TN	true negative.
TP	true positive.

References

[AMAZ17] Albawi S., Mohammed T. A., Al-Zawi S.: Understanding of a convolutional neural network. In *2017 International Conference on Engineering and Technology (ICET)* (2017), pp. 1–6. doi:10.1109/ICEngTechnol.2017.8308186. 2

[Con19] Constantinou V.:. Hey-waldo [online]. 2019. URL: https://github.com/vc1492a/Hey-Waldo [cited 2021-06-17]. 3

[DT05] Dalal N., Triggs B.: Histograms of oriented gradients for human detection. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)* (2005), vol. 1, IEEE, pp. 886–893. doi:10.1109/CVPR.2005.177. 2

[DX17] Deng J., Xie X.: Nested shallow CNN-cascade for face detection in the wild. In *2017 12th IEEE International Conference on Automatic Face Gesture Recognition (FG 2017)* (2017), pp. 165–172. doi:10.1109/FG.2017.29. 3

[Han87] Handford M.: *Where's Wally*. Walker Books, London, United kingdom, 1987. 2, 3

[HZRS16] He K., Zhang X., Ren S., Sun J.: Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), pp. 770–778. doi:10.1109/CVPR.2016.90. 3

[KB14] Kingma D. P., Ba J.: Adam: A method for stochastic optimization. *3rd International Conference for Learning Representations, San Diego, 2015 8*, 1 (2014), 1–15. arXiv:1412.6980. 4

[KKSK20] Kaur P., Krishan K., Sharma S. K., Kanchan T.: Facial-recognition algorithms: A literature review. *Medicine, Science and the Law 60*, 2 (2020), 131–139. PMID: 31964224. doi:10.1177/0025802419893168. 2

[Kot07] Kotsiantis S. B.: Supervised machine learning: A review of classification techniques. In *Emerging Artificial Intelligence Applications in Computer Engineering*, Maglogiannis I., Karpouzis K., Wallace B. A., Soldatos J., (Eds.). IOS Press, Oct. 2007, pp. 3–24. 2

[LLS*15] Li H., Lin Z., Shen X., Brandt J., Hua G.: A convolutional neural network cascade for face detection. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015), pp. 5325–5334. doi:10.1109/CVPR.2015.7299170. 3

[SWA19] Sae-Lim W., Wettayaprasit W., Aiyarak P.: Convolutional neural networks using mobilenet for skin lesion classification. In *2019 16th International Joint Conference on Computer Science and Software Engineering (JCSSE)* (2019), vol. 16, pp. 242–247. doi:10.1109/JCSSE.2019.8864155. 3

[SWH18] Sun X., Wu P., Hoi S. C.: Face detection using deep learning: An improved faster rcnn approach. *Neurocomputing 299* (2018), 42–50. URL: https://www.sciencedirect.com/science/article/pii/S0925231218303229, doi:https://doi.org/10.1016/j.neucom.2018.03.030. 3

[VJ01] Viola P., Jones M.: Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'01)* (2001), vol. 1, pp. 511–518. doi:10.1109/CVPR.2001.990517. 2

[ZZR*20] Zheng Y., Zhao Y., Ren M., Yan H., Lu X., Liu J., Li J.: Cartoon face recognition: A benchmark dataset. In *Proceedings of the 28th ACM International Conference on Multimedia* (New York, NY, USA, 2020), MM '20, Association for Computing Machinery, p. 2264–2272. URL: https://doi.org/10.1145/3394171.3413726, doi:10.1145/3394171.3413726. 4