Where's Waldo: Image Recognition with Random Forest, Convolutional Neural Network, and Support Vector Machine

Alberto Avitia
CSU, Long Beach
GitHub: betoDevs
`albertoavitia.d@gmail.com`

Darren Ly
CSU, Long Beach
GitHub: lydarren
`dxddarren@gmail.com`

Daniel Wang
CSU, Long Beach
GitHub: danqo
`danioo.w@gmail.com`

Abstract

Image classification has become a popular challenge in machine learning. In this experiment the challenge of finding "Waldo" within a given image was tackled by using three popular machine learning models: Random Forest, Support Vector Machine, and Convolutional Neural Network. Each model was trained with 64X64 samples in a classification manner. The models were then tested by using a 64x64 "sliding window" on the bigger image and applying it's predictions on each "window" in order to find a locate Waldo. Issues arised due to the low amount and imbalance of the data. The results for this problem indicate that Convolutional Neural Networks provided the best accuracy, while Support Vector Machine was favored in terms of speed. Random Forest produced the best results with minimal data preparation and augmentation while also being fast. The results gives insight to the advantages and disadvantages between different the machine learning models for image recognition.

1 Introduction

"Where's Waldo" is a series of children's puzzle books that challenges the reader to find a specific character in an illustration. The challenge arises in first training a model to correctly classify "Waldo", and second, find "Waldo" within a large, noisy image. The purpose of this experiment is to the test and compare three machine learning models in the context of image recognition: Random Forest, Convolutional Neural Network and Support Vector Machine.

2 The Dataset

The original dataset contains 19 full sized RGB images broken down into 64x64 sub-images. 39 images are classified as "Waldo" and 5451 images are classified as "Not Waldo" that can be found on kaggle [1]. More images were found online and books leading to a total of 57 positive Waldo to train.

Every dataset contains a number of issues. The most problematic issue with this dataset were the size of the images, the size of the object we wanted to classify, and the amount of noise in the images.

There is about an 18:1 ratio of negative vs positive. The size of the actual character took up about 10% of the positive images. Considering the size of the images used to train were 64x64 RGB, each actual Waldo was about a 10x10 RGB when testing.

3 Methodologies

3.1 Random Forest

Random Forest were chosen for this model because of its ability to overcome overfitting, dealing with outliers [6], handling missing data and are able to train in a faster time, when compared to other machine learning models. Random Forests is an ensemble machine learning model that groups together Classification and Regression Trees (CART) [9] in order to make multiple votes on the same data [7][8]. The most voted decision among the trees is then chosen as the prediction.

3.1.1 Random Forest Model Details

Being that classification was needed for classifying Waldo, only Classification Trees were chosen to form part of the Random Forest model (RF). RandomForestClassifier from the sklearn, Pandas and Numpy libraries were used in Python 3.6. The RF was formed by 60 Decision Trees and each decision tree within the RF was allowed to grow to full depth. Bootstrap aggregating, or Bagging, in junction with random sampling was used in the creation of the trees. Entropy and information gain was used for the splitting criterion at each node. During the testing phase the predict_proba function was used to determine a positive value if the probability was higher than 70%. The pipeline for the creation and training of the RF can be seen in Figure (1).

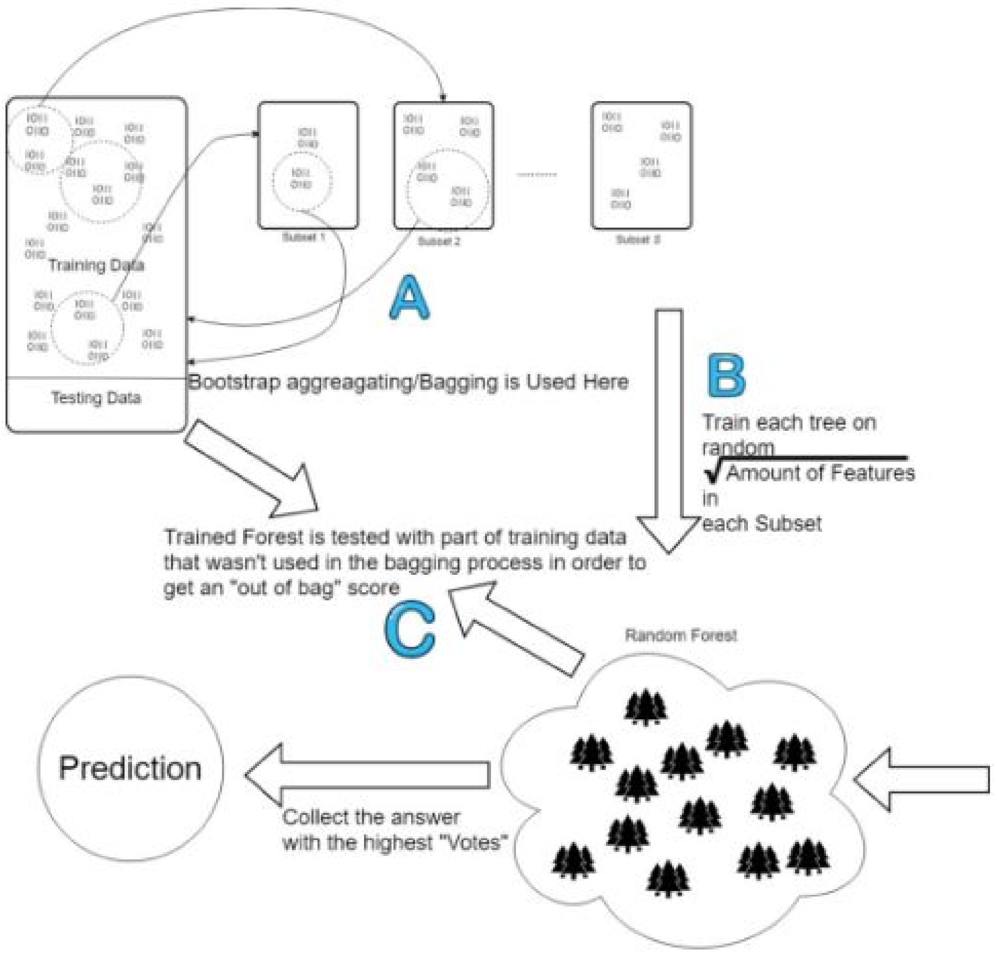

Training Data
Testing Data
Subset 1
Subset 2
Subset 3
A
Bootstrap aggreagating/Bagging is Used Here
B
Train each tree on random √Amount of Features in each Subset
Trained Forest is tested with part of training data that wasn't used in the bagging process in order to get an "out of bag" score
C
Random Forest
Prediction
Collect the answer with the highest "Votes"

Figure 1: Pipeline of bagging and training of a Random Forest Algorithm

3.1.2 Bootstrap Aggregating and Training

Bootstrap aggregating [14] refers to the random sampling, with replacement, in order to create subsets of the original dataset D. Here, D is split into two subsets: training dataset, and testing dataset. The following process will refer to Figure (1). During (A) Bagging, data k from the training dataset is picked at random and placed into a Subset s, then randomly picks data m, where m < k, and places it back in the training dataset. This process is repeated until the training dataset is depleted. (B) Each subset is then used to create one tree by randomly picking $\sqrt{AmountofFeatures}$ values from the subset and using entropy and information gain in order to split at the node and train the tree. This process is followed until the set amount of trees are created. (C) The testing data is now given to the RF and each tree determines an output. The most common output gets chosen as the prediction, and is compared to the label corresponding to the testing data. The score from this testing phase is given as the "out-of-bag score". This Model was trained on a total of 1005 64x64x3, where each image was flattened into an numpy array and a 1 or 0 was attached to determine the label for "Waldo" and "not Waldo". The RF was tested in the same way, by having a flattened 64x64x3 array which resulted in a 1X12288 array and being inputted to the model.

3.1.3 Undersampling

It is worth noting that the data used for this experiment on the RF was composed of 560 positives and 5451 negatives. According to literature[15], the predictions will tend to focus more on the prediction accuracy of the majority class, which will result in poor accuracy for the minority class. The method to overcome this problem was to undersample negative class by randomly choosing 545 samples and training the model them.

3.2 Convolutional Neural Network (CNN)

Convolutional Neural Networks are often used for image recognition. Two convolutional neural networks were used that were trained on the dataset with a batch size of 64, and a learning rate of 0.01 for 100 epochs using Google Colaboratory. The first model's configuration follows that of the VGG with 16 layers [4]. The difference is the usage of the standard SGD, an input of size 64x64, and an output layer of 2 units. The second model's configuration follows that of the first model, but a dropout of p = 0.5 is applied to each pooling layer and a dropout of p = 0.3 is applied between the first two fully connected layers. Figure (2) shows the second model's configuration.

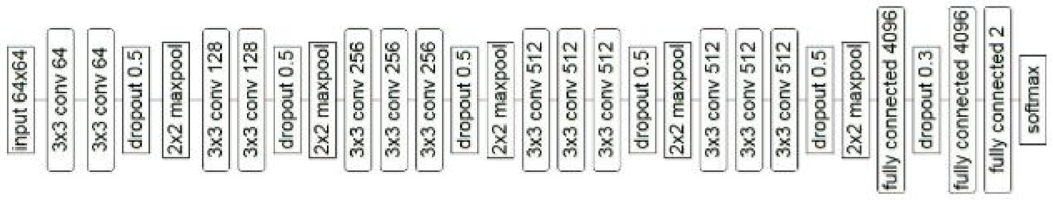


Figure 2: CNN model configuration

3.2.1 Oversampling

A second copy of the positive "Waldo" images were added to the training dataset to combat the class imbalance problem. By increasing the amount two times, from 700 to 1400 total images. One problem that can arise from oversampling is overfitting the model. Validation accuracy may not reflect those in the testing accuracy due to the possibility of validating on an image that appears in both the training and validation set.

3.2.2 Maxpool Dropout to Reduce Overfitting

Max pool dropout refers to applying a dropout to the max pooling layer which provides a way to seemingly pick a feature from the feature map randomly based on the multinomial distribution, similar to that of stochastic pooling[4][5]. A dropout probability of p=0.5 was chosen based on the optimal probability chosen from Hinton's dropout paper and the results from Hu's max pool dropout comparison [2][3]. Max pool will keep the high value features from the map even if it is unneeded information, whereas average pooling will use all the useless and useful values from the feature map. To compromise by applying max pool dropout to the noisy images during training, there is a possibility of removing the unneeded features containing high values that max pool and average pooling will use.

3.2.3 Extra Data Augmentation to Reduce Overfitting

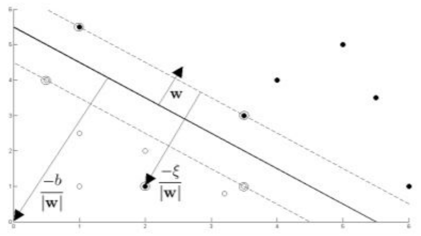
Extra images were needed to decrease the overfit of negative "Waldos" by trying to balance the number of classes. More positive Waldo images were created by shifting the images horizontally and vertically by 20% of the image size which also leads to more noise. By shifting the images so that Waldo can appear in different locations of the images, the neural network would learn more features pertaining to the positive Waldo from different locations. 560 additional images were created leading to a total of 700 positive "Waldo" to train. With the oversampling done on the training set, this increases the number of positive Waldo images to 1400.

3.3 Support Vector Machine (SVM)

Support Vector Machine is a supervised learning model that attempts to classify data by creating a separating hyperplane, much like the Perceptron Learning Algorithm. However, unlike Perceptron, it also attempts to find the best hyperplane. The closest members of both classes to the separating hyperplane are known as the Support Vectors. The distances between the separating hyperplane and the Support Vectors is known as the margin. The secondary goal of the Support Vector Machine is to maximize the distance from Support Vectors. In the case where the data is not linearly separable, there is a parameter C that determines the trade-off value between misclassification and margin size [12].

$$\min \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{L}\xi_i \quad \text{s.t.} \quad y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1 + \xi_i \geq 0 \ \ \forall_i$$

(a) SVM objective function



(b) SVM graphical representation

3.3.1 Histogram of Oriented Gradients (HOG) Feature Extraction

The histogram of oriented gradients is the chosen method of extracting features from the images. The purpose of this feature descriptor is to reduce the amount of features necessary for computation while keeping features that describe or visualize the structure of an image [11]. It does this by computing the distribution of edge directions.

First, the horizontal and vertical gradient images are computed by filtering the image with kernels. The gradient images are used to compute the magnitude and orientation of the gradients. Next, the image is divided into blocks of cells and the histogram of gradients for each block. Lastly, the histogram of oriented gradients is normalized and used to construct a feature vector that describes the edges in the image [13].

The original training images of size 64x64 with RGB channels would have required the SVM classifier to compute with 12288 features. With HOG, the number of features with the chosen parameters was reduced to 5292 features. The configuration of 9 orientations, 8 pixels per cell, and 2 cells per block was chosen. A configuration using 12 orientations and 16 pixels per cell was tested but proved to be less accurate. A configuration using 12 orientations and 8 pixels per cell resulted in comparable accuracy despite having 7056 features. The feature extraction was done using all three color channels.

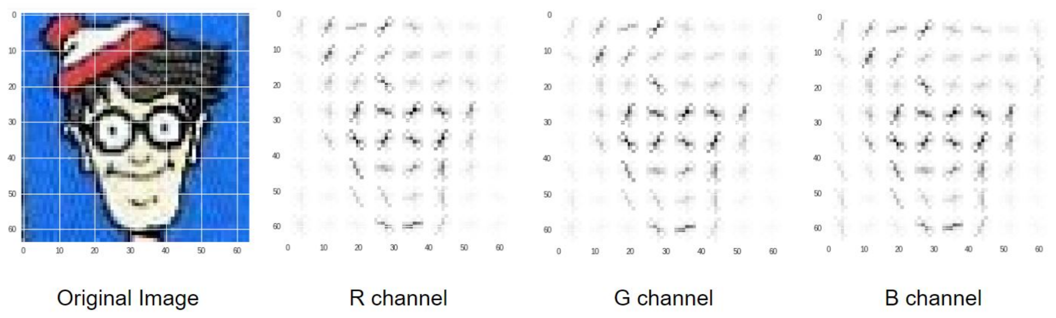


Figure 4: Example of HOG visualization

3.3.2 Support Vector Machine Classifier Details

The procedure of using the histogram of oriented gradients as a feature descriptor and the Support Vector Machine as a classifier was inspired by Dalal and Triggs [10]. L2 norm regularization penalty is known to be standard for the Support Vector Machine, since L1 norm leads to sparser coefficients. Both regularization techniques achieved very similar accuracy results, but using L1 norm was significantly faster to train. The squared hinge loss was used instead of the hinge loss because it provided better accuracy.

3.4 Sliding Window

A sliding window algorithm was devised to search for Waldo in the full images, which were much larger than our 64 x 64 sized training data. The sliding window starts at the top leftmost corner of the picture and slices the Each window is 64x64, just like the training data. The necessary feature extractions were performed, if any, and the classifier is applied on the window. The window is then shifted over, and the process is repeated for the entire image.

4 Results

Greyscale or white-masked tiles indicate "not Waldo", while colored tiles indicate "Waldo".

4.1 Random Forest Results

Out-of-Bag Score for the RF model was 76.7% and a train time of 5.69 seconds, but unfortunately waldo was not found on the following images and there was a great number of false positives.

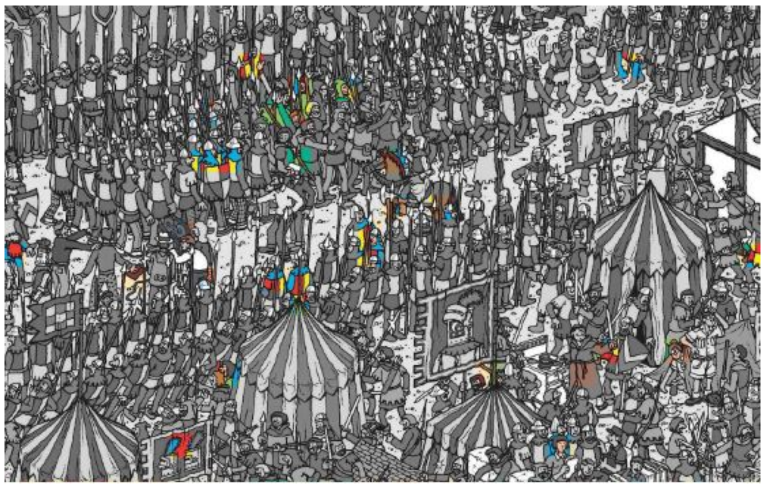


Figure 5: Result on full-sized image #63 using Random Forest classifier

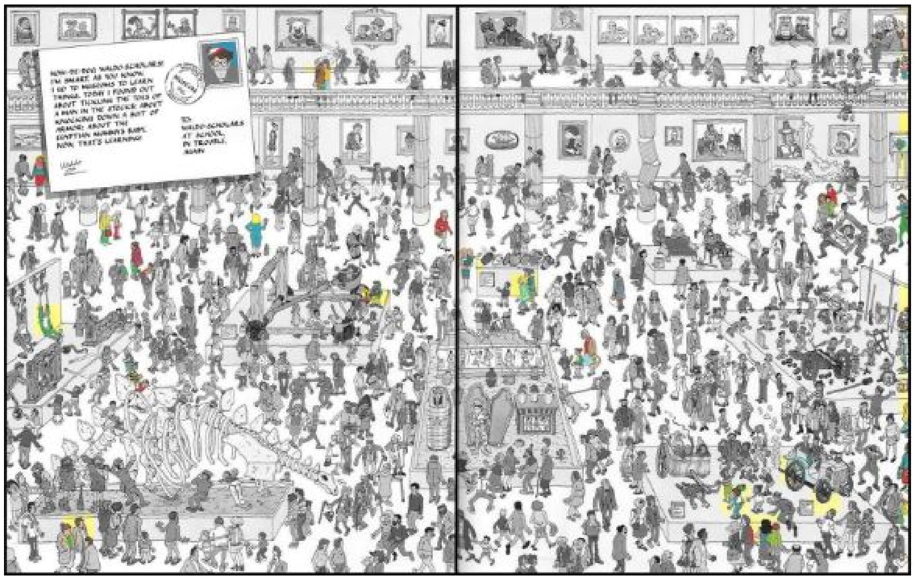


Figure 6: Result on full-sized image #74 using Random Forest classifier

4.2 CNN Results

The Convolutional Neural Network achieved high accuracy on the 64x64 training images, but found several false positives on the full-sized images. In addition, it took the longest to train: upwards of 30 minutes for each model. Nonetheless, it achieved the lowest number of false positives on the full-sized images while still being able to find Waldo.

4.2.1 VGG Results

The VGG architecture found Waldo along with false positives:

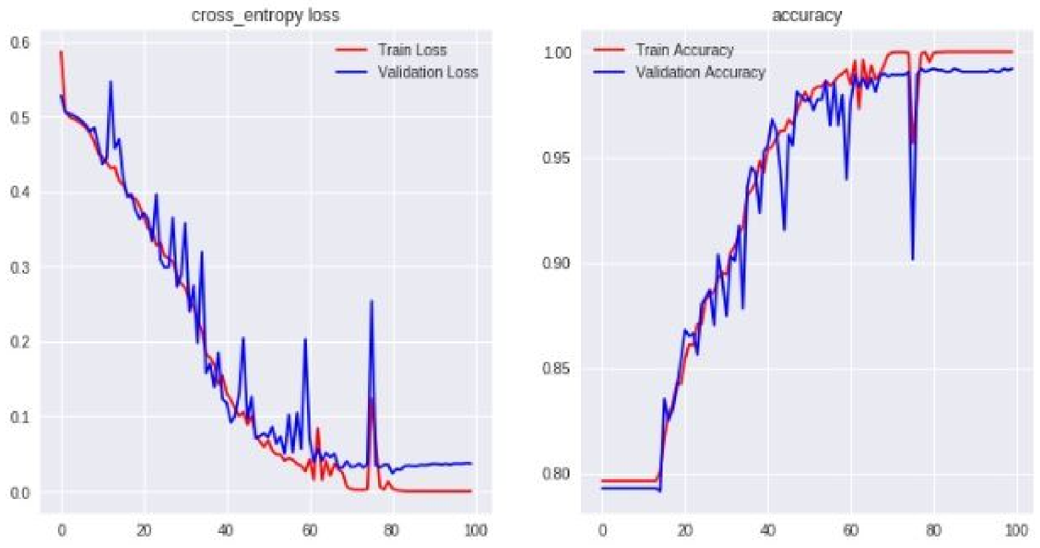


Figure 7: VGG cross entropy loss and accuracy graphs

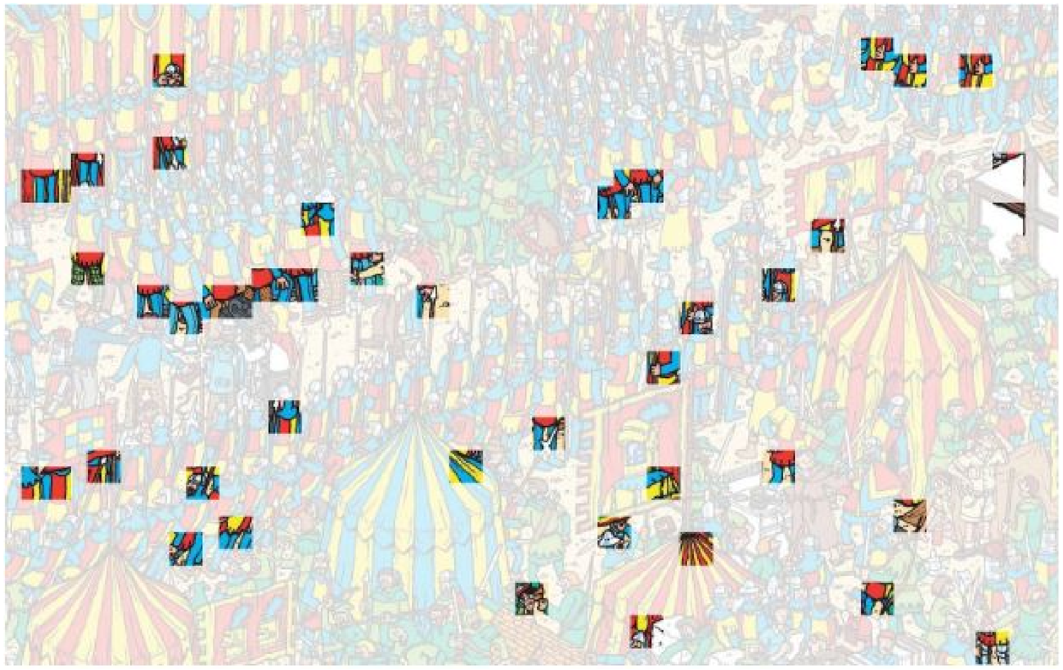


Figure 8: Result on full-sized image #63 using VGG

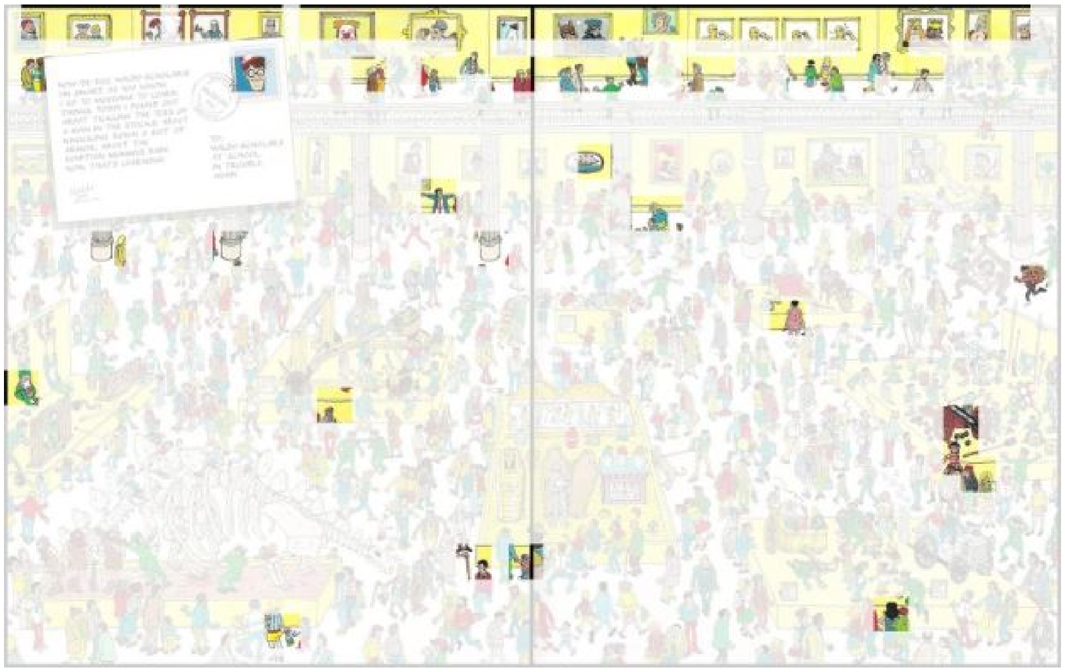


Figure 9: Result on full-sized image #74 using VGG

4.3 Maxpool VGG Results

The Maxpool VGG variant was able to reduce the number of false positives while still finding Waldo:

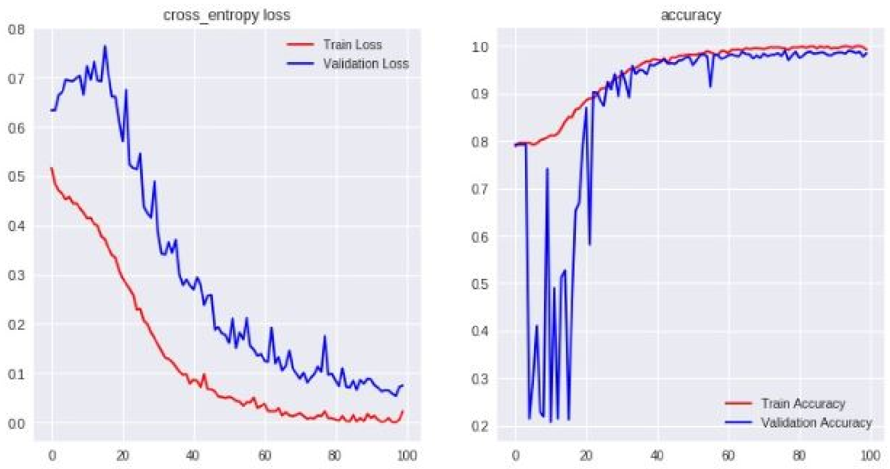


Figure 10: Maxpool VGG cross entropy loss and accuracy graphs

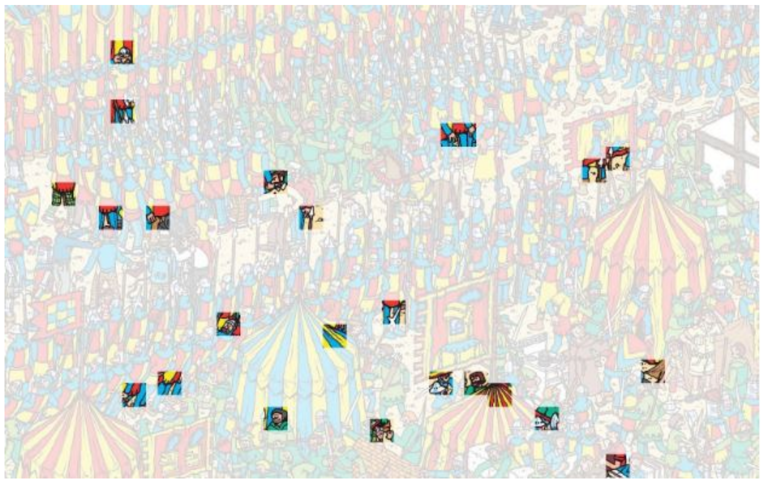


Figure 11: Result on full-sized image #63 using Maxpool VGG

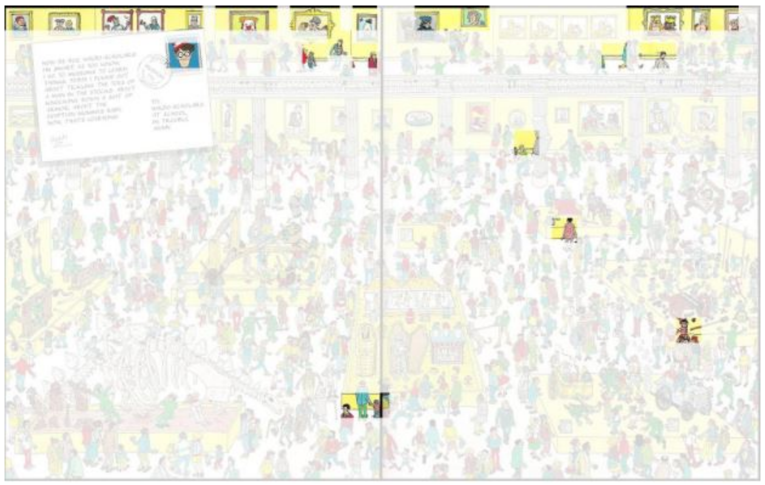


Figure 12: Result on full-sized image #74 using Maxpool VGG

4.4 Support Vector Machine Results

Support Vector Machine achieved 99.87% training accuracy and 98.21% validation accuracy. It also had very quick training times: 2.31 seconds with L1 norm and 7.80 seconds with L2 norm. Both regularization However, it had much more false positives compared to the Convolutional Neural Network models.

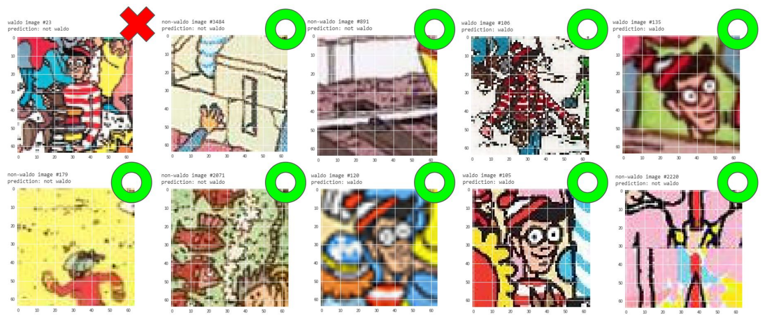


Figure 13: Random samples on dataset HOG feature extraction and SVM classifier

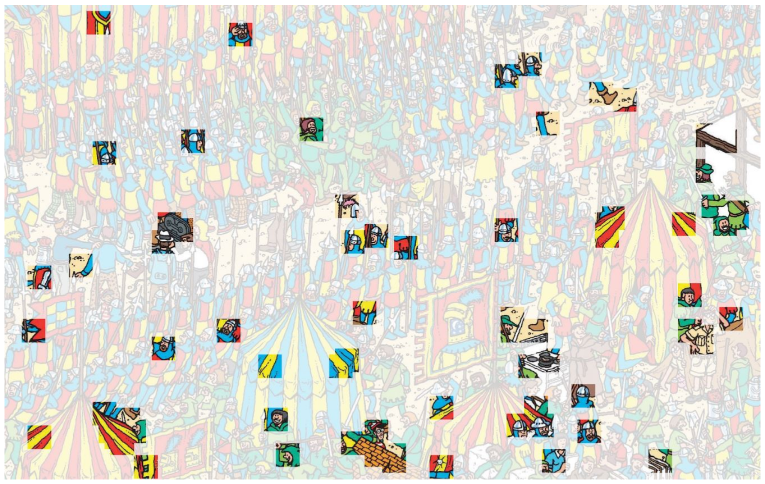


Figure 14: Result on full-sized image #63 using HOG feature extraction and SVM classifier

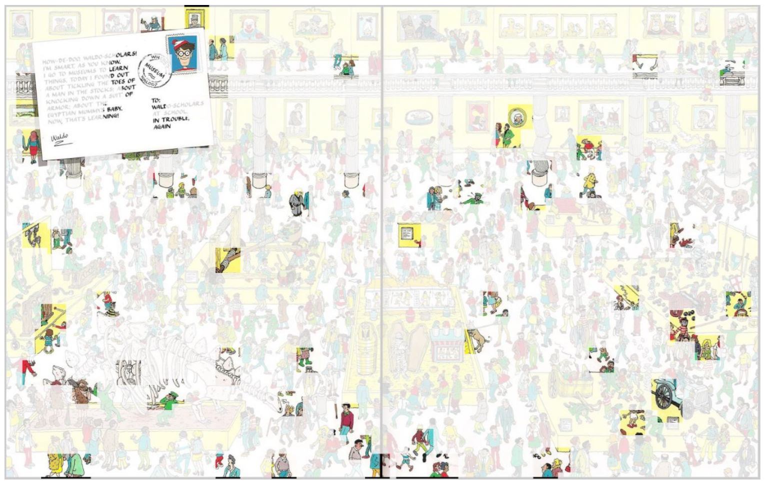


Figure 15: Result on full-sized image #74 using HOG feature extraction and SVM classifier

5 Conclusion

Three models were trained and used to predict and determine the location of a "Waldo" image within a bigger image.The models trained were Random Forest, Convolutional Neural Network, and Support Vector Machine. The dataset to train the models was very low and thus augmentation methods were applied to the data in order to increase the number of samples. Convolutional Neural Network was the model with the best results by being able to detect Waldo and produce the least amount of false positives. Support Vector Machine was also successful in identifying Waldo and produced only a slightly higher number of false positives. Random Forest was not successful in fully identifying Waldo, only part of him, but was able to perform better than the other two models with limited data preparation and augmentation. Hopefully the results gives some insight to the advantages and disadvantages between the different machine learning models for image recognition.

6 References

[1] Alexander, J.A. & Mozer, M.C. (1995) Template-based algorithms for connectionist rule extraction. In G. Tesauro, D.S. Touretzky and T.K. Leen (eds.), *Advances in Neural Information Processing Systems 7*, pp. 609–616. Cambridge, MA: MIT Press.

[2] Bower, J.M. & Beeman, D. (1995) *The Book of GENESIS: Exploring Realistic Neural Models with the GEneral NEural SImulation System.* New York: TELOS/Springer–Verlag.

[3] Hasselmo, M.E., Schnell, E. & Barkai, E. (1995) Dynamics of learning and recall at excitatory recurrent synapses and cholinergic modulation in rat hippocampal region CA3. *Journal of Neuroscience* **15**(7):5249-5262.

[1] A. Bilogur. Where's Waldo Kaggle Dataset. 25-Oct-2017. [Online]. Available: https://www.kaggle.com/ residentmario/wheres-waldo.

[2] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. 2014. [Online]. Available: https://www.cs.toronto.edu/ hinton/absps/JMLRdropout.pdf

[3] H. Wu, X. Gu. Max-Pooling Dropout for Regularization of Convolutional Neural Networks. 2015. [Online]. Available: https://arxiv.org/abs/1512.01400

[4] K. Simonyan, A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. 2015. [Online]. Available: https://arxiv.org/abs/1409.1556

[5] M. D. Ziler, R. Fergus. Stochastic Pooling for Regularization of Deep Convolutional Neural Networks. 2013. [Online]. Available: https://arxiv.org/abs/1301.3557

[6] J. Ali, R. Khan, N. Ahmand, I. Masqood. Random Forests and Decision Trees. IJCSI International Journal of Computer Science. 2012. [Online]. Available: https://pdfs.semanticscholar.org/959a/8e906ee26b940374b719 253c8e188ed78fd3.pdf

[7] N. Horning. Random Forests: An algorithm for image classification and generations of continuous fields and data sets. American Museum of Natural History, Center for Biodiversity and Conservation. 2012. [Online]. Available: http://wgrass.media.osaka-cu.ac.jp/gisideas10/papers/04aa1f4a8beb619e7fe711c29b7b.pdf

[8] L. Breiman. Random Forests. University of California Berkeley, Statistics Department. 2001. [Online]. Available: https://www.stat.berkeley.edu/ breiman/randomforest2001.pdf

[9] W. Log. Classification and regression trees. John Wiley & Sons, Inc. 2011. [Online]. Available: http://www.stat.wisc.edu/ loh/treeprogs/guide/wires11.pdf

[10] N. Dalal and B. Triggs. Histograms of Oriented Gradients for Human Detection. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR05).*

[11] C. Vondrick, A. Khosla, T. Malisiewicz, and A. Torralba. HOGgles: Visualizing Object Detection Features. *2013 IEEE International Conference on Computer Vision*, 2013.

[12] T. Fletcher. Support Vector Machines Explained. 01-Mar-2009. [Online]. Available: http://sutikno.blog. undip.ac.id/files/2011/11/SVM-Explained.pdf

[13] S. Mallick. Image Recognition and Object Detection: Part 1. Learn OpenCV, 14-Nov-2016. [Online]. Available: https://www.learnopencv.com/image-recognition-and-object-detection-part1/

[14] L. Breiman. Bragging Predictors. University of California Berkeley, Department of Statistics. 1994. [Online]. Available: https://statistics.berkeley.edu/sites/default/files/tech-reports/421.pdf

[15] C. Chen, A. Liaw, L. Breiman. Using Random Forest to Learn Imbalanced Data. University of California Berkeley, Department of Statistics. 2004. [Online]. Available: http://statistics.berkeley.edu/sites/default/files/tech-reports/666.pdf