Where's Waldo?

Sunghwan Baek

Jong-Ik Park

18752 Project

Content

- Introduction
- Feature Extraction
- Binary Classification
- Exploring Alternative Methods
- Conclusion

Why we chose the "Finding Waldo" topic

1. First, it presents a fun and engaging way to demonstrate the power of machine learning algorithms in solving real-world pattern recognition problems.

2. Second, finding Waldo in a crowded scene resembles common challenges in computer vision, such as object detection and recognition, making it an excellent test case for exploring various machine-learning techniques.

3. Lastly, the popularity of the "Where's Waldo?" series generates interest and accessibility for a broad audience, allowing us to showcase the potential of machine learning in a relatable and entertaining context.

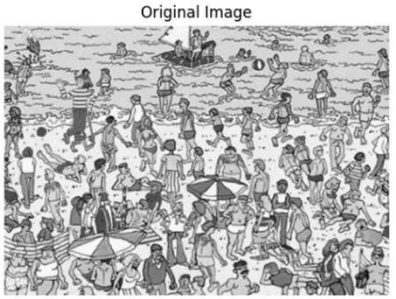
Dataset – Waldo and Wilma Dataset

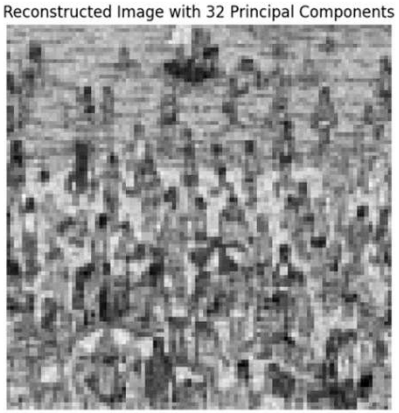
- The objective of the game is to find Waldo in a busy and chaotic scene filled with various characters and objects.
- It IS challenging and time-consuming.
- The complexity of the scene and the similar appearance of the characters make it difficult to locate Waldo.
- The dataset consists of 10000 images (5000 images with dimensions 350x500 and 5000 with dimensions 224x224) and the 2 characters (Waldo and Willma) which are required to find in the images

https://www.kaggle.com/datasets/sheshngupta/waldowilma

Principal Component Analysis (PCA)

PCA method didn't work well…

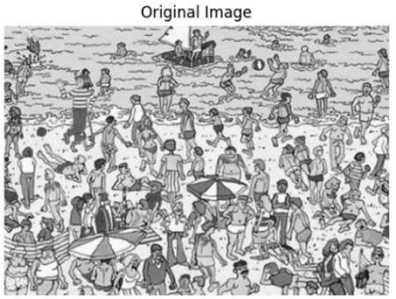


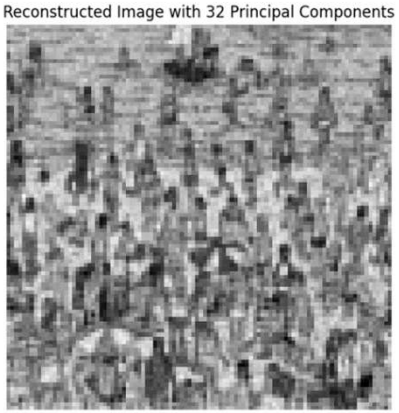


- PCA, or Principal Component Analysis,
- is a **dimensionality reduction technique** commonly used in machine learning and data analysis.
- The main idea is to **transform the original high-dimensional data** into **a lower-dimensional space** while **preserving most of the data's information** (i.e., variance).
- The new dimensions, called principal components, are linear combinations of the original features and are orthogonal to each other.

Principal Component Analysis (PCA)

Inappropriate for object detection

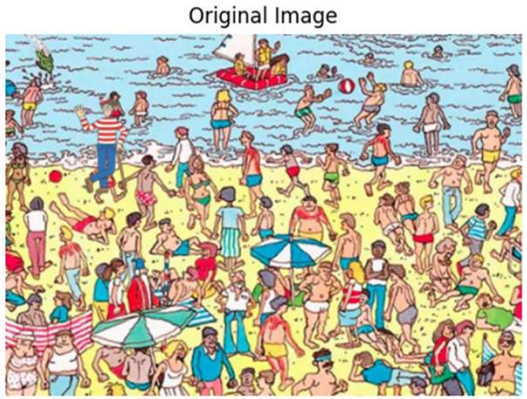


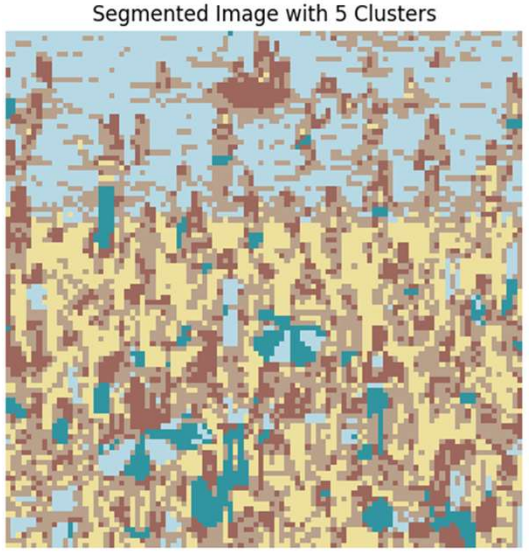


- PCA is a dimensionality reduction technique data and reducing noise.
- It is **not designed for object detection tasks** like finding Waldo, which require the identification of specific patterns and features in the image.
- As a result, the PCA-transformed images might not contain the information needed for successful detection.

K-mean clustering didn't work well…

- K-means clustering is an unsupervised machine learning algorithm
- Used to partition data points into a specified number of clusters (k) based on their similarity.

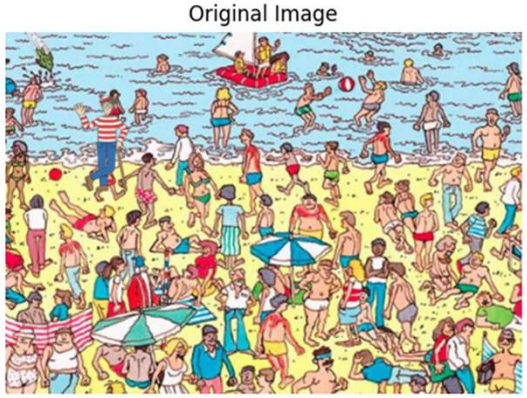


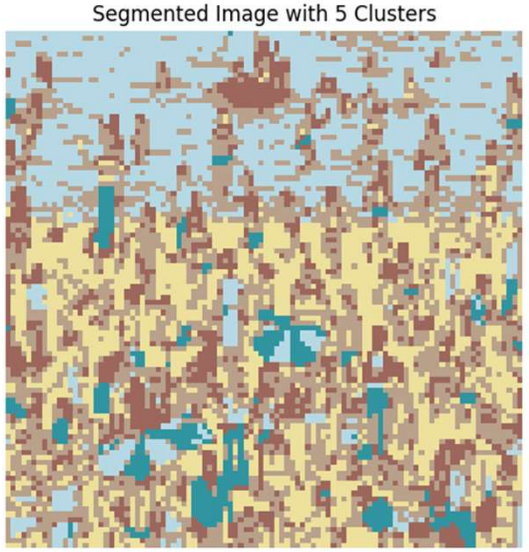


K-means Clustering

Assumes equal-sized clusters

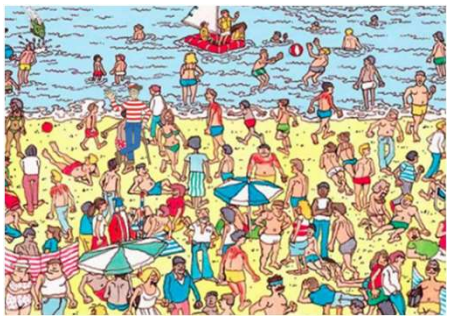
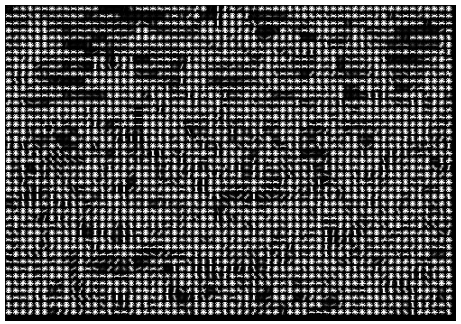
- K-means clustering assumes that the clusters are roughly equal in size, which might not hold in the "Finding Waldo" problem.
- Waldo is a small object in a large and complex scene, and using K-means clustering may result in inappropriate cluster assignments, leading to poor object detection performance.



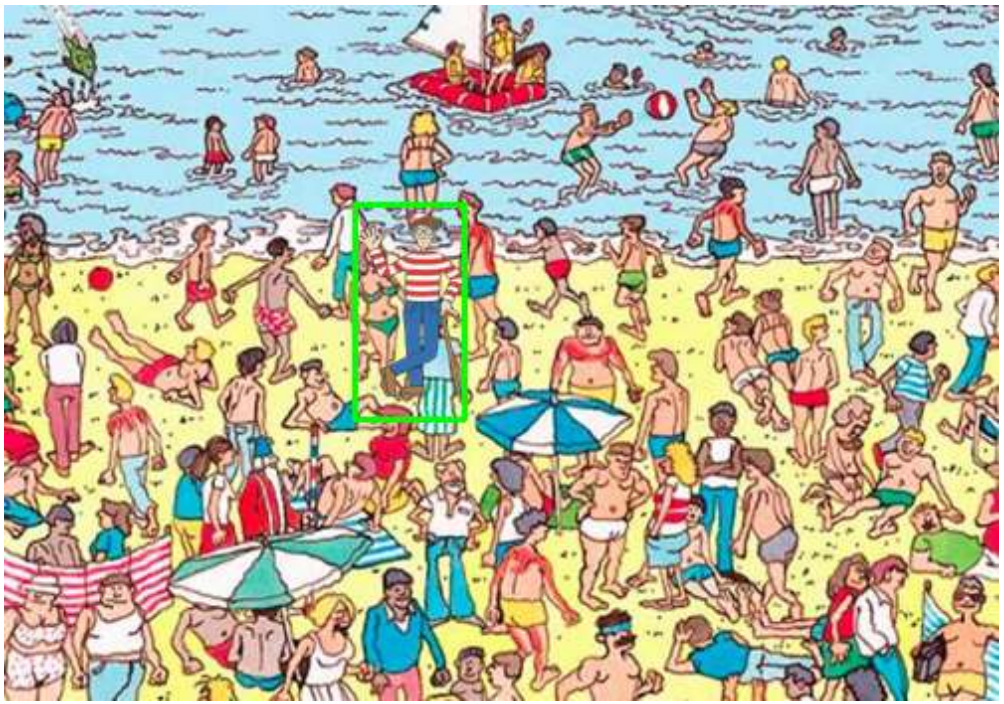


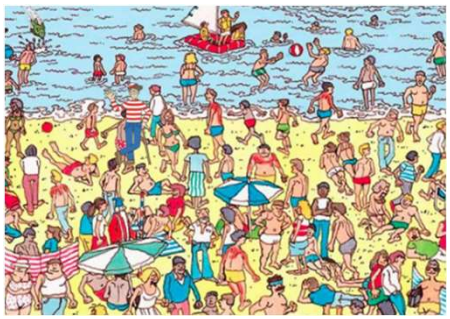
It work well!

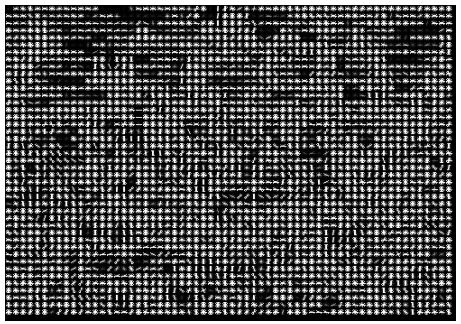
- The Histogram of Oriented Gradients (HOG) is a popular feature extraction technique used in computer vision and image processing tasks, such as object detection and recognition.
- The primary idea behind HOG is to **capture the shape and appearance** of an object **by analyzing the distribution** and **orientation of intensity gradients** in an image.


Histogram of Oriented Gradients (HOG)







Problem

We needed to use binary classification.

- In the "Finding Waldo" task,
  one challenge encountered when using HOG
  as a feature descriptor is the presence of visually similar characters,
  such as Wilma, who might share similar HOG features with Waldo due to their close resemblance.

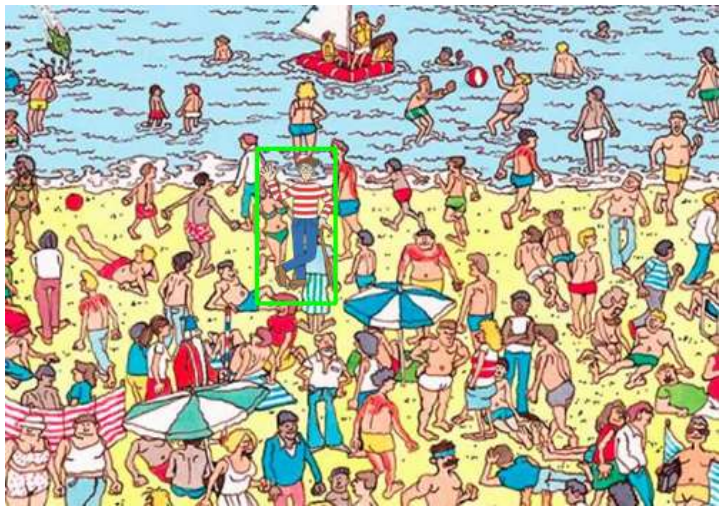
- This issue can lead to false positives, where the algorithm mistakenly identifies Wilma as Waldo.

Steps for the solution(Binary)

1. Extract HOG features for both the input image and the Waldo template.

2. Perform template matching using HOG features of the input image and the Waldo template.

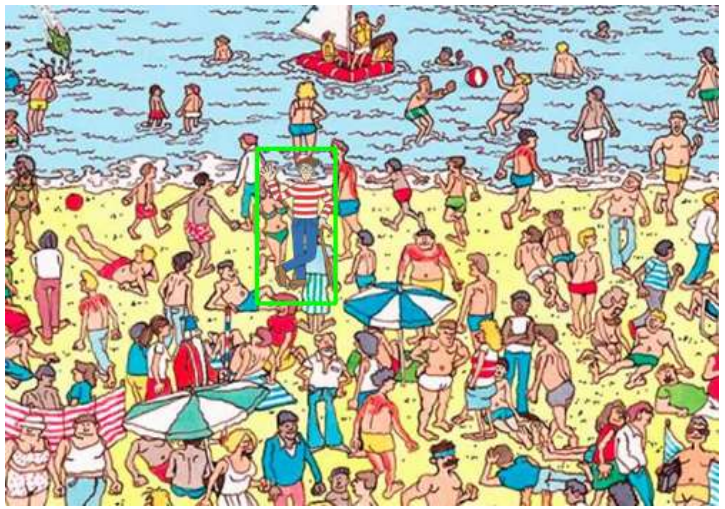
3. Calculate the Intersection over Union (IoU) between the predicted and ground truth bounding boxes. If the IoU is greater than or equal to 0.5, it suggests that the matched section is correct.



Steps for the solution(Binary)

4. Calculate the difference between the HOG descriptors of the matched sub-image and the Waldo template. we have used the L2-norm to compute the difference between the HOG features. If this difference is less than or equal to the threshold it indicates that the matched character is Waldo. Otherwise, the matched character is considered different.

5. Iterate through all the images in the dataset, applying the find_waldo function and counting the number of correct identifications.
The overall accuracy of the model is then calculated and reported.



Using Waldo template to find Waldo

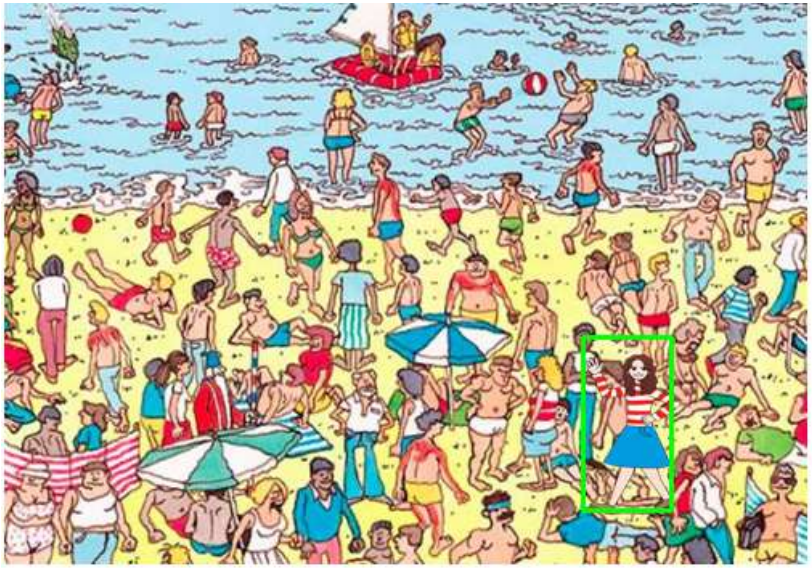
The accuracy = 100%

- We used 30 different background images and the codes detected where the waldo is with the accuracy of 100%. This task showcases the ability of our machine learning model to effectively recognize a specific target amidst a complex and cluttered environment.

Classifying non-Waldo(Wilma) image

Our code classifies with 100% that it is not Waldo

- we utilized the Waldo template to classify and differentiate Wilma within 30 background images.
- By successfully identifying that the images containing Wilma did not feature Waldo, we demonstrated the model's robustness and effectiveness in distinguishing between characters with similar appearances.
- This highlights the model's capability to prevent false-positive detections and accurately locates Waldo in complex and cluttered scenes.
- (In the image, it shows 0% accuracy, and it means non of them are Waldo and that is correct!



HOG overlay on the Waldo template:

```
HOG descriptor of the main image:
[0.19688879 0.         0.         ... 0.24454351 0.0853706  0.

HOG descriptor of the Waldo template:
[0. 0. 0. ... 0. 0. 0.]
19.31838087498843
HOG difference: 19.31838087498843 (exceeds threshold)
This is a different character!
Accuracy: 0.00%
```

Binary Classification using alternative methods

We separated our images into Waldo and non-waldo(Wilma).

```
if (foldername=='waldo'):
        labels.append(1)
    else:
        labels.append(0)
```

- Then…We train our model using 4 different methods

Other methods

Logistic Regression Accuracy: 57.50% (It is not good!)

```
# Logistic Regression
lr = LogisticRegression()
lr.fit(X_train, y_train)
y_pred_lr = lr.predict(X_test)
lr_accuracy = accuracy_score(y_test, y_pred_lr)
print(f"Logistic Regression Accuracy: {lr_accuracy * 100:.2f}%")
```

Support Vector Machine Accuracy: 56.25% (It is not good!)

```
# Support Vector Machine
svm = SVC()
svm.fit(X_train, y_train)
y_pred_svm = svm.predict(X_test)
svm_accuracy = accuracy_score(y_test, y_pred_svm)
print(f"Support Vector Machine Accuracy: {svm_accuracy * 100:.2f}%")
```

Random Forest Classifier Accuracy: 57.50% (It is not good!)

```
# Random Forest Classifier
rf = RandomForestClassifier()
rf.fit(X_train, y_train)
y_pred_rf = rf.predict(X_test)
rf_accuracy = accuracy_score(y_test, y_pred_rf)
print(f"Random Forest Classifier Accuracy: {rf_accuracy * 100:.2f}%")
```

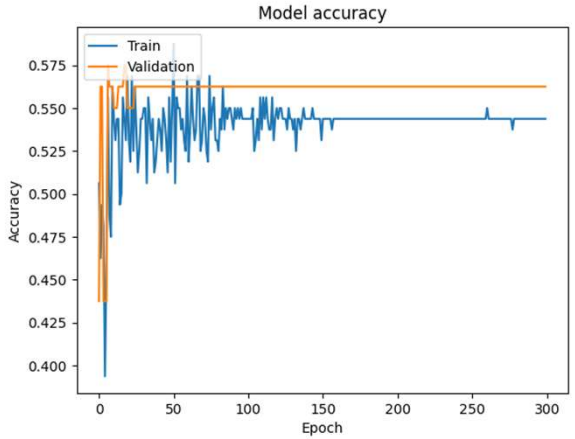
Other methods

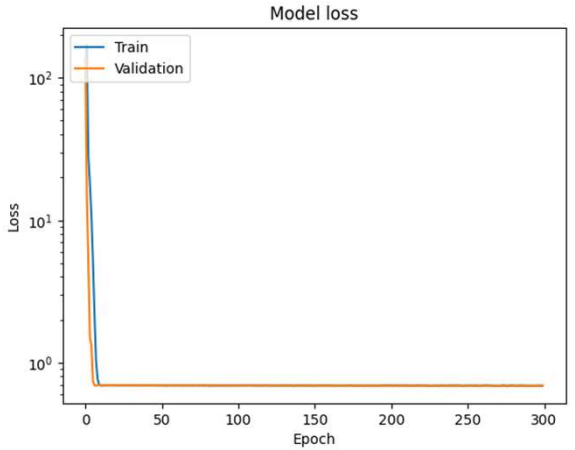
Convolutional Neural Network Accuracy: 56.25% (It is not good!)

```
# Convolutional Neural Network
X_train_cnn = X_train.reshape(X_train.shape[0], 128, 128, 1)
X_test_cnn = X_test.reshape(X_test.shape[0], 128, 128, 1)

cnn = Sequential()
cnn.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(128, 128, 1)))
cnn.add(MaxPooling2D(pool_size=(2, 2)))
cnn.add(Dropout(0.25))
cnn.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
cnn.add(MaxPooling2D(pool_size=(2, 2)))
cnn.add(Dropout(0.25))
cnn.add(Flatten())
cnn.add(Dense(128, activation='relu'))
cnn.add(Dropout(0.5))
cnn.add(Dense(1, activation='sigmoid'))
cnn.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
history = cnn.fit(X_train_cnn, y_train, batch_size=128, epochs=300, verbose=0, validation_data=(X_test_cnn, y_test))

y_pred_cnn = (cnn.predict(X_test_cnn) > 0.5).astype("int32")
cnn_accuracy = accuracy_score(y_test, y_pred_cnn)
plot_training_graph(history)
print(f"Convolutional Neural Network Accuracy: {cnn_accuracy * 100:.2f}%")
```





Our code worked better for finding Waldo

- We thought the Waldo template is like a key and fingerprints.

- And since we know the pattern, of the Waldo, it was better to just compare to the image in the background.

Now, we designed our code to…

1. Performed template matching using the HOG features of both Waldo and Wilma templates and computed the locations with the highest similarity.

2. Extracted matched sub-images for both Waldo and Wilma, computed their HOG features, and calculated the difference between the HOG descriptors.

3. We classified the image as either Waldo or Wilma based on the more negligible difference between HOG descriptors.

Back to the original algorithm

Now, we designed our code to…

```
        # Extract the matched sub-image
        matched_sub_image_waldo = image[top_left_waldo[1]:bottom_right_waldo[1],
                                  top_left_waldo[0]:bottom_right_waldo[0]]
        matched_sub_image_willma = image[top_left_willma[1]:bottom_right_willma[1],
                                  top_left_willma[0]:bottom_right_willma[0]]


        # Calculate the HOG descriptor for the matched sub-image
        matched_sub_image_waldo_features, _ = get_hog_features(matched_sub_image_waldo, **hog_params)
        matched_sub_image_willma_features, _ = get_hog_features(matched_sub_image_willma, **hog_params)

        # Calculate the difference between the HOG descriptors
        hog_diff_waldo = np.linalg.norm(matched_sub_image_waldo_features - waldo_template_feature)
        hog_diff_willma = np.linalg.norm(matched_sub_image_willma_features - willma_template_feature)

        # If the difference between the matched sub-image and Waldo's template is smaller
        # than the difference between the matched sub-image and Wilma's template, classify the image as Waldo (1).
        # Otherwise, classify it as Wilma (0).

        if hog_diff_waldo > hog_diff_willma:
          prediction.append(0)
        else:
          prediction.append(1)


    return prediction
```

We used 240 images

```
[ ]  # Calculate accuracy
     accuracy = accuracy_score(labels, prediction)
     print(f"Accuracy: {accuracy * 100:.2f}%")

     Accuracy: 100.00%
```

Our algorithm classified Wilma and Waldo image with the accuracy of 100%!

Conclusion

Why the original algorithm worked better?...

- HOG captures the object's shape and edge information well, which is essential when identifying characters like Waldo and Wilma, who have distinct appearances.

- Using a template-matching approach, the algorithm explicitly searches for the features present in the templates (Waldo and Wilma).
  This allows the algorithm to focus on the most relevant information for the classification task, making it more accurate than methods that need to learn the relevant features from scratch.

Conclusion

Why the original algorithm worked better?...

- Because our dataset is relatively small,
  more is needed to effectively train a CNN,
  which usually requires large amounts of data to achieve high performance.

- HOG and template matching can work better in such cases, as they rely on hand-crafted features that are already known to be effective in detecting the target objects.

Conclusion

Why the original algorithm worked better?...

- Our best model was using

  HOG feature extraction with l2 norm

  to find the difference + Binary classification.

- Other classification combined difference method, but they didn't work well with low accuracy.

Thank you for listening!