Guess Who?

Introducing Waldo: an Application for Facial Recognition and OCR

Andrew Gibbs-Bravo

INM460 Computer Vision, MSc Data Science
School of Mathematics, Computer Science and Engineering
City, University of London

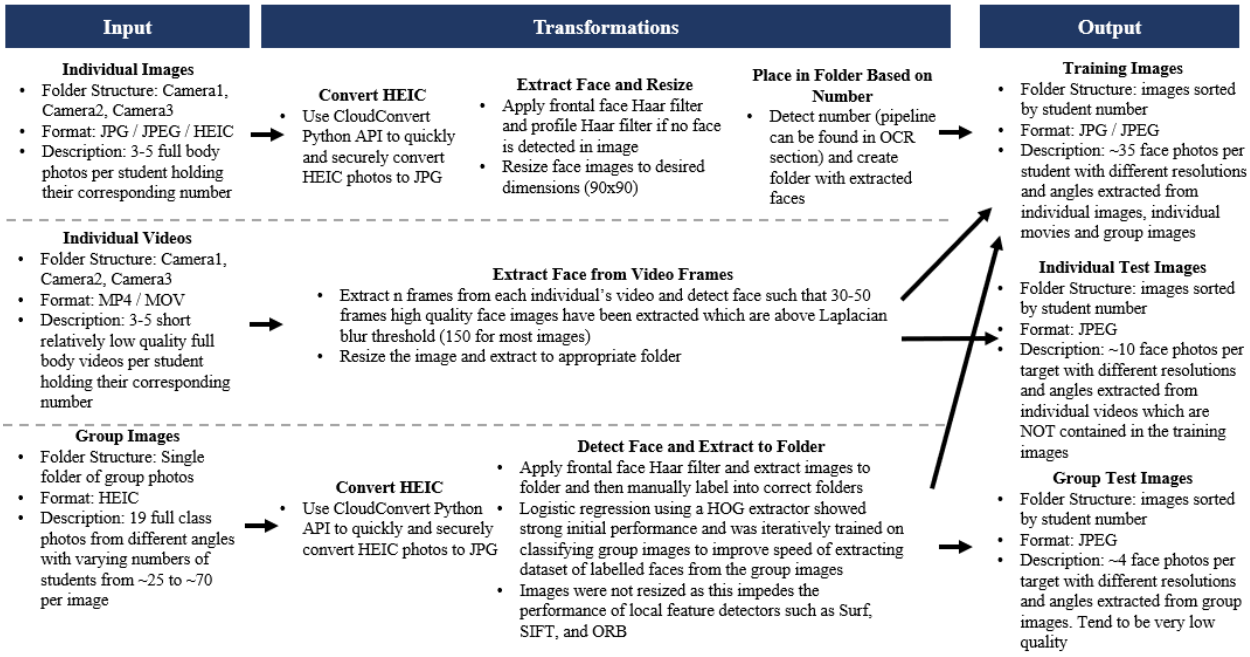
Abstract

This paper introduces Waldo, an efficient and scalable application for facial recognition and optical character recognition on both group and individual photos and video. The preprocessing, training, and testing pipeline will be discussed in addition to the underlying theory. Waldo's strong performance on unseen individual and group images will be demonstrated in addition to its ability to accurately detect numbers. Areas for future work and the implications of facial recognition applications are also discussed.

1 Project Overview

Facial recognition technology is becoming increasingly prevalent with applications to criminal identification, payment authorization, advertising, and access and security among others. The main functionality Waldo provides is the ability to detect one or more faces in an image and recognize the ID (similar to name) which it corresponds to. IDs are generated based on the number the target is holding and therefore Waldo performs optical character recognition (OCR). Waldo's strong facial recognition performance can be attributed to its data preprocessing pipelines as facial recognition requires a reasonable number of high-quality training images which are similar to the target images. This paper demonstrates the well-known idea that training data is a more significant driver of performance than algorithm choice and as such, the various processing pipelines are the main contribution to the strong results.

Preprocessing Pipeline

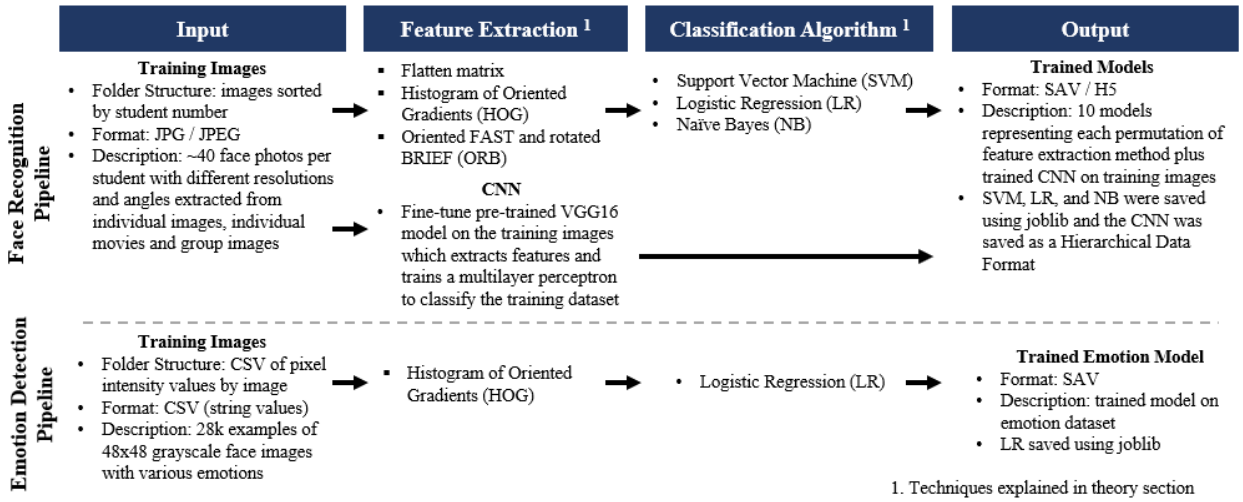


As described above, the preprocessing pipeline converts the raw images from full body and group images in various formats to a set of labeled folders by ID containing only the extracted faces from each image. The major folders which result from this are:

- Two training folders which are combined into a trainset dataframe consisting of 60% of the photos sampled from the individual training images plus all the group images:
    - Individual training folder: contains faces extracted from the individual images and individual videos. Only 60% are randomly selected as the photos are highly correlated to one another therefore it increases computation time but not performance
    - Group training folder: contains faces extracted from group photos. All group photos were used (except photos used for testing)
- Two testing folders which contain unseen images and are used individually for model evaluation:
    - Individual test folder: contains unseen frames from generated from the individual videos
    - Group test folder: contains unseen extracted faces from five of the group photos

This was a very time-consuming process as automating it essentially involved iteratively completing the project. In order to get the images in their folders I first extracted the face from each image and used the OCR pipeline to map it to its corresponding folder ID. I then trained a model on the individual faces which I used to predict the extracted group photos which became the training set. The initial performance was quite poor and required significant manual correction although as the training set became populated with more group photos, the labelling accuracy improved dramatically as the model was retrained resulting in the strong performance on the unseen group test set. Although automation was time-consuming, my hard drive crashed during the project and resulted in the loss of the folders which would have required a significant time investment to recreate if done manually. Automating the process allowed me to recreate the folder structure very quickly and would allow the dataset to scale easily to include new faces. Using a model to label the group training images also helped label images which I had trouble assessing visually.

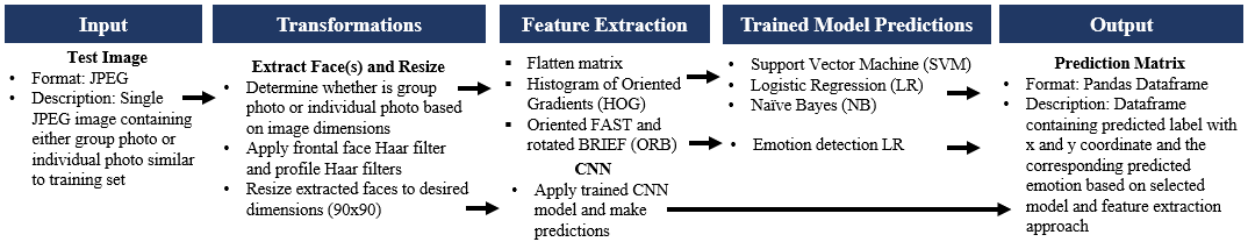
Model Training Pipeline



Once the training and test sets were created, I trained the face recognition models for comparison on the test sets. As detailed above, each model is trained using each feature extraction technique to determine which combination achieves the best performance. As CNN performs feature extraction and classification end-to-end it does not require a feature extraction approach to transform the raw pixel data. The trained models are then exported for use in deployment.

The second pipeline is for emotion detection and required preprocessing in order to train a classifier. The dataset is called FER 2013 and consists of ~28.7k images of facial expressions with labels (Carrier et al., 2013). Only happy, sad, angry, and surprised images were used, as specified in the requirements document. Images were stored as strings therefore I needed to convert to NumPy integer arrays and reshape into 48x48 images. The images were then resized into 90x90 so the HOG feature dimensions would remain consistent with face detection pipeline.

Model Prediction Pipeline



The RecogniseFace function in Waldo follows the above model prediction pipeline. Waldo will load the trained models as specified by the key parameters: featureType and classifierName. When ORB is selected as featureType, Waldo will also load the trained Kmeans classifier which is used in constructing the visual bag of words. The trained emotion detection logistic regression model is applied to the extracted HOG features. Predictions are compiled with their corresponding x and y coordinates into the prediction matrix output.

1.1 Programming Environment and Libraries

Waldo has a Python backend to provide a scalable and efficient application which can also leverage deep learning models. Minimizing the required dependencies without compromising performance was a high priority in order to ensure no runtime errors occur. As such, very few external libraries were used:

- CloudConvert API: conversion of HEIC files to JPG
- OpenCV (CV2): main image processing library in Python
- Tesseract / PyTesseract: OCR library developed by Google with pretrained models for high performance
- Numpy / Scipy / Pandas / Scikit-learn: matrix manipulation, training algorithms, and handling dataframes
- Keras / TensorFlow: fine-tuning of pretrained CNN (VGG16) and use of autoencoders
- OS / shutil: creating and manipulating folder structures and file names

2 Overview of Algorithms & Theory

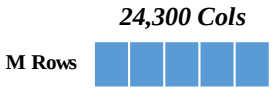
2.1 Feature Extraction Approaches

Feature extraction approaches are designed to transform the raw pixel intensity data into a meaningful feature vector which a classification algorithm can be trained on. Good features are those which are invariant to scale, shift, and rotation and are robust to noise.

2.1.1 Flattened

Flattening the pixel intensity matrix of an image is more of a preprocessing approach to ensure that the model can be trained on vectors of consistent length although it represents a good benchmark to compare feature extraction approaches to. The approach involves creating a single array from the pixel intensities and loses locality information aside from the adjacent pixels in a row.

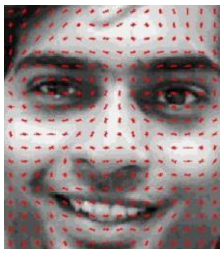
Flattened Array Dimensions



2.1.2 Histogram of Oriented Gradients (HOG)

HOG feature descriptors are created using the distribution of the gradients (forms a histogram of the directions of the pixel intensities) on a grayscale image for image detection through identifying edges and distinct shapes. This approach was introduced by Dalal and Triggs (2005) and consists of segmenting an image into localized portions and calculating the gradient and magnitude for that portion. This is done using a sliding window approach and block normalization is performed to improve the lighting invariance of the features. The next step is to create a global histogram of the occurrence of the different gradients by angle degree bins. This is similar to the bag-of-visual-words concept in SURF / ORB (which is discussed

HOG Feature Example

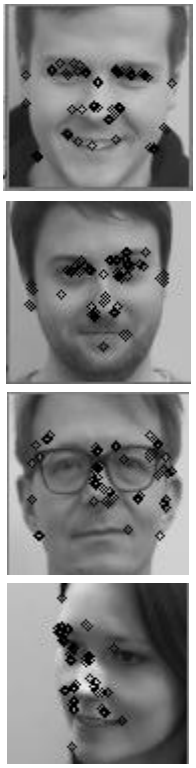


Source: Thaware 2018

below), however rather than having multiple descriptors per image there is only a single feature vector per image which is the flattened HOG vector.

2.1.3 SURF / SIFT / ORB

SURF Feature Examples

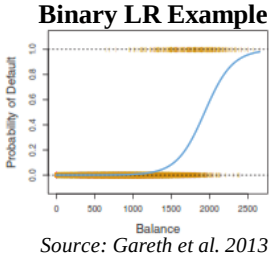


SIFT and SURF and similar feature extraction approaches which both rely on composing a histogram of meaningful local features that represent the image. Both approaches are scale invariant, rotation invariant, and illumination invariant although they are under patent and therefore not included in the base OpenCV package. As of the objectives in developing Waldo is to it achieve strong performance while minimizing dependencies, OpenCV's Oriented Fast and Rotated BRIEF (ORB) algorithm was used instead. ORB combines FAST keypoint detection to identify keypoints with some modifications and BRIEF to describe the keypoints. To determine the orientation, ORB computes the intensity weighted centroid of the cell and uses the direction of the vector. To improve rotation invariance ORB has a moment threshold for the keypoints and introduces an adjustment to BRIEF which 'steers' it in the same orientation as the keypoints (Rublee et al. 2011).

One of the issues with extracting local features is that there are inconsistent numbers extracted per image. To resolve this, an approach from NLP is borrowed which involves quantizing the features and representing the images using a histogram referred to as constructing a visual bag of words (VBOW). Waldo uses a K-means clustering algorithm with a dictionary size of 500 to quantize the features. In its original implementation, Waldo used a default K-means implementation although the computation speed was far too slow. Therefore, a mini batch K-means implementation was used instead which improved speed considerably and did not appear to reduce prediction accuracy. Training the models on the BOVW dictionary afterwards was also much faster than either the HOG or flattened approaches as the matrix was only 500 dimensional. As seen in the example images, the features which ORB detects tends to focus on the eyes, nose, and mouth as they contain the greatest number of corners.

2.2 Classification Algorithms

Binary LR Example



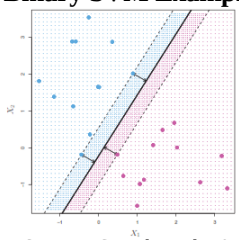
Source: Gareth et al. 2013

2.2.1 Logistic Regression

Logistic regression is a very common linear classification model which typically applies a sigmoid function to transform predictions to probabilities rather than real valued estimates as is common in linear regressions. The cost function used in Waldo is categorical cross-entropy (also referred to as log-loss) and is the most common function for multiclass classification problems. Logistic regression also has variants which aim to regularize the model to improve generalization to unseen data. The two most common approaches are lasso regression and ridge regression although neither approach was used in Waldo's logistic regression implementation.

2.2.2 *Support Vector Machine (SVM)*

Binary SVM Example

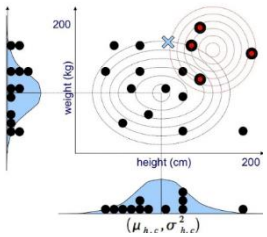


Source: Gareth et al. 2013

SVMs are similar to logistic regressions in that they try to find a hypersurface that separates classes. The approach is different in that it tries to maximize the distance to the nearest training data point therefore the only points which influence the boundary are the support vectors. The main advantage to support vector machines is the "kernel trick" which permits a nonlinear kernel (such as polynomial or radial-basis-function) that transforms the feature space so a linear boundary can be formed in multidimensional space. As this can result in significant computational intensity, Waldo only applies a linear kernel.

2.2.3 *Gaussian Naïve Bayes*

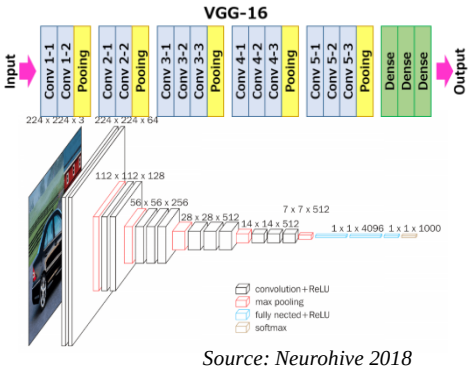
Gaussian NB Example



Source: Lavrenko 2015

Gaussian naïve bayes is a member of the naïve bayes family of models which assume independence between features and predict based on the joint probability of the feature distribution. Naïve bayes is common in natural language processing which uses binary or categorical variables (presence of words in documents) and is not widely used in image recognition. The discrete model is adapted to the continuous case by using maximum likelihood estimation of a parameterized distribution for each feature in this case using a gaussian distribution. As the model assumes independence between features it is not expected to perform well in image recognition.

2.2.4 *Convolutional Neural Networks (CNN)*



Source: Neurohive 2018

CNNs are deep neural networks which combine a series of transformations to pixel data to extract higher level features including convolving filters across an image, max pooling, fully connected layers and dropping out certain activations randomly. There are many CNN architectures which combine these and other components although Waldo implements the VGG16 architecture (shown to the right). One of the greatest benefits of CNN models is that model weights which are trained on large datasets such as ImageNet can be fine-tuned to classify similar images not contained in the training set using transfer learning. Waldo fine-tunes the pre-trained VGG16 model by removing the final dense feedforward multilayer perceptron layers and retraining a single fully connected layer on the face images which reduces training time and increases accuracy.

3 Face Recognition – Analysis & Evaluation of Results

3.1 Evaluating Algorithmic Performance | Data is King

In determining which model performs the best, four different classification algorithms were tested in addition to three different feature extraction approaches. The best performing model across all holdout test datasets is a trained logistic regression model using a HOG feature extractor. Logistic regression also has the lowest average model file size in terms of memory of the approaches which is potentially important for use cases requiring smaller models such as cellphones.

Model Performance Across Training Datasets

	60% of Individual Photo Data						60% of Individual Photos + Extracted Group Photos					
	Test Dataset						Test Dataset					
Model	Individual			Group			Individual			Group		
CNN	99.0%			35.3%			99.4%			95.0%		
Feature Extraction	**HOG**	**ORB**	**Flatten**	**HOG**	**ORB**	**Flatten**	**HOG**	**ORB**	**Flatten**	**HOG**	**ORB**	**Flatten**
SVM	99.3%	94.8%	98.6%	20.2%	7.9%	8.3%	99.4%	94.7%	98.7%	94.6%	51.5%	88.8%
Logistic Regression	99.7%	95.4%	99.4%	63.0%	46.5%	36.1%	99.7%	95.1%	99.3%	95.9%	49.0%	91.7%
Naïve Bayes	97.7%	76.1%	93.6%	7.9%	2.1%	10.0%	96.7%	63.2%	93.1%	50.6%	10.4%	36.9%
Average	**98.9%**	**88.8%**	**97.2%**	**30.4%**	**18.8%**	**18.1%**	**98.6%**	**84.3%**	**97.0%**	**80.4%**	**36.9%**	**72.5%**

The CNN Waldo applies is a fine-tuned VGG-16 model which achieves good accuracy although fails to generalize to the group images when only training on the individual photo data. The CNN achieves competitive performance on both the individual test dataset when the extracted group photos are included. Experimentation with different architectures and hyperparameters would likely result in improved performance although this architecture was selected as it is simple and trains quickly while achieving strong performance.

As the SVM has a linear kernel it performs somewhat similarly across feature extraction methods and testing datasets to logistic regression as both are linear classifiers. However, logistic regression far outperforms SVM on group test accuracy using only 60% of the individual photo data which indicates that it generalizes far better to unseen cases. The Gaussian naïve bayes classifier was expected to perform quite poorly as it assumes features are independent from one another which clearly does not hold in image data even with feature extraction techniques.

HOG achieved the best performance of the feature extraction techniques although flattening the pixel intensities was surprisingly effective. The performance of ORB was very poor on group images even when faces extracted from different group image photos were included in the training set. Given the competitive performance of ORB on the individual test dataset there does not seem to be an error in the implementation, and I suspect the poor performance is a result of image resolution and quality. ORB is scale invariant as it uses image pyramids. However, during testing I noticed a significant improvement in classification accuracy using ORB if the testing images are all rescaled to 90x90 pixels. This suggests that the images are too low quality to produce reliable ORB features as the group test images have many faces that are very low resolution. Many approaches were tried to improve ORB's performance including trying different implementations and different preprocessing approaches on the images.

The main driver of performance however is not the choice of algorithm but the data which is used to train the algorithm. Average HOG performance increased from 30.4% to 80.4% when faces extracted from different group images were added to the training set. The performance of every algorithm increased dramatically with SVM using a flattened feature extractor increasing the most from 8.3% to 88.8%.

This again demonstrates the importance of ensuring that the data which the model is using to train matches that which it will make predictions on.

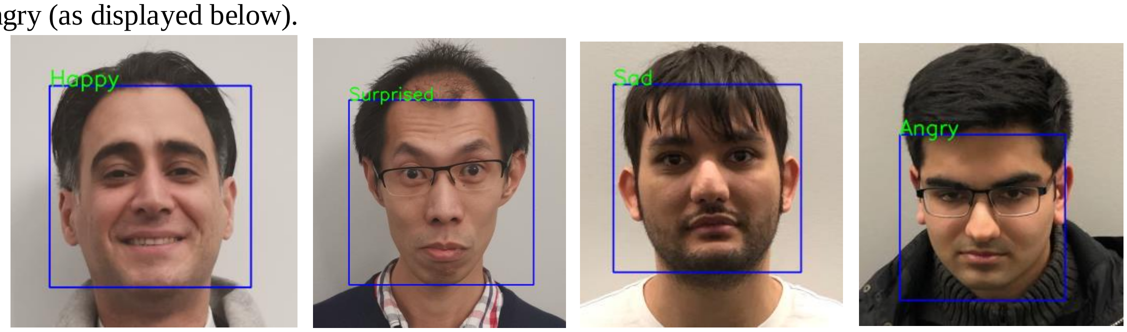
Example of Waldo Performance on Group Photo

Waldo Achieves 100% Accuracy on Detected Faces

The face recognition component of Waldo achieves very high performance on the group holdout test set and even better on the individual holdout test set. The face detection component however has room for improvement. As seen in the first image above, Waldo misses one completely visible face and another which is only partially obstructed. This results from Waldo's reliance on Haar filters which are discussed in the failure cases section and future work section. Faces which are not labeled in the training set are denoted as other and are not included in the matrix which Waldo outputs.

3.2 Emotion detection

Waldo is also capable of detecting the emotions of target faces and whether they are happy, surprised, sad or angry (as displayed below).



In order to achieve this, Waldo is trained on a subset of a previous Kaggle competition dataset for facial expression recognition called FER 2013 (Carrier et al., 2013). Only faces which displayed one of the above emotions were included in the training set per the requirements document. The model which was used was the same architecture as the best performing face recognition model: a logistic regression classifier with a HOG feature extractor. Only a subset of the data was used to improve computation time and the model's performance on the Kaggle test set was not very impressive achieving only ~60% accuracy therefore there is definite room for improvement. Other libraries were considered and tested including: EmoPy and Emotion although they required too much overhead and dependencies to meet one of the key project objectives.

3.3 Waldo is Robust to False Positives

Face detection is the first step in facial recognition and one of the vulnerabilities of Waldo is its reliance of Haar filters which rely on simple filters rather than more complex and higher-performing deep learning face detection approaches. Haar filters are susceptible to false positives in environments with high numbers of edges and shading (Viola 2001). As such, the face detection portion of Waldo's face recognition pipeline was tested on the three images that do not contain faces. No faces were detected on the first two images although the third image had a high number of false positives (highlighted squares). Through the application of a second Haar filter all but two false positives are eliminated and of the two only one was classified as a face.

Waldo Tested on Landscape Images to Check for False Positives

The red squares in the third image are rejected false positive faces
and the blue squares are the ones which passed the second filter.

3.4 Unsuccessful Approaches

Several other approaches were attempted in facial recognition including:

- VGG-Face: pretraining from a Keras model which was trained on the 'Faces in the Wild' dataset rather than ImageNet. This was assumed to result in better performance given the weights would likely be more adapted to the nuances in faces however the model failed to improve and achieved poor performance (~60% accuracy) on the individual test dataset. It is possible this was due to a mistake in implementation or insufficient data to fine-tune the weights.
- Face_recognition library: a python library for out of the box facial recognition. This is a popular library with over 23,000 stars on Github although it had a high number of dependencies including requiring a C++ compiler which conflicts with the project aims. The performance in face detection was equivalent to using a Haar filter when tested on group photos therefore it was not adopted into Waldo.
- Applying autoencoders and deep autoencoders: in an effort to improve the performance of models trained only on the individual face photos on the group face photos I attempted to apply

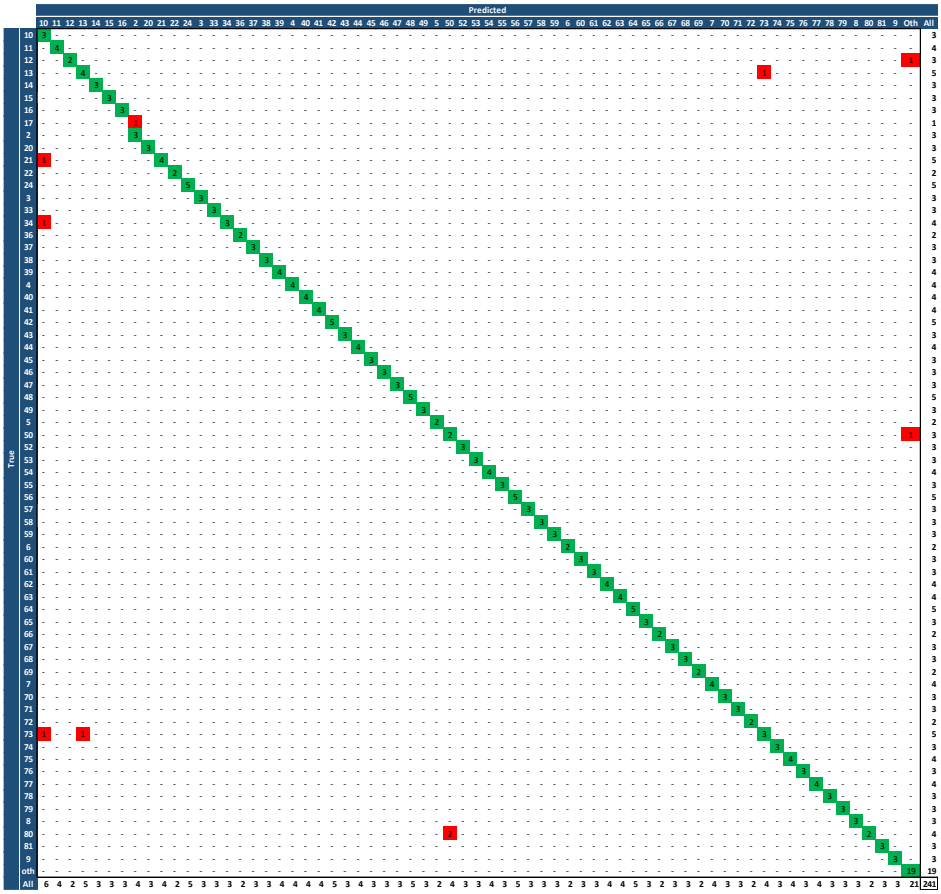
autoencoders to reduce the dimensionality of the feature extraction approaches (e.g. HOG) and then apply LightGBM, a strong non-linear model. This unfortunately did not perform competitively as there was insufficient data to adequately train an autoencoder (or a deep autoencoder).

- Use a GAN to standardize images: Again, in an effort to improve the performance of models trained only on the individual face photos and tested on the group face photos I considered applying a GAN to convert the low-quality group face images into the better-quality individual face images. The transformed images would then be fed to the trained classifier. This approach was not implemented as the transformations would require too much computation to be practical.

3.5 Confusion Matrix and Misclassified Faces

Waldo achieves very strong performance on the group holdout test set using the best performing model (logistic regression with HOG feature extraction) and the misclassifications are reasonable.

Waldo Confusion Matrix Performance on Group Holdout Test Set (95.6%)



Note: predicted is sample image selected from training set as only labels are predicted

3.6 Failure Cases

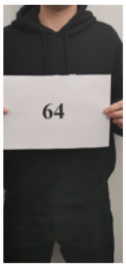
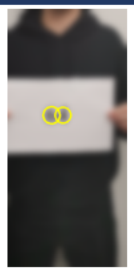
Waldo Failure Example

Waldo's main vulnerability in facial recognition is the initial step which is face detection. As Waldo relies on Haar filters, faces which have poor lighting conditions, are obstructed, or are only partially in the image are often missed and therefore not classified by the recognition pipeline. Similarly, for the individual photos, photos which are taken on too sharp an angle above or below the target are often not recognized as Waldo only uses frontal and profile face filters. Other vulnerabilities include poor generalization as the images which Waldo used to train on had relatively high correlation to one another in that they occurred with relatively similar resolution, angles, and lighting conditions. As demonstrated in the poor model performance when only training on the individual images and predicting on the group images, the model performance depends largely on the data which it was trained on. The model is unlikely to contain sufficient training data to be able to perform well on images which are significantly different from the training set such as images where the target is wearing sunglasses or a moustache.

4 Optical Character Recognition (OCR) – Analysis & Evaluation of Results

Optical character recognition is a key feature offered by Waldo and was used in the preprocessing pipeline in constructing folders grouped by target ID number. Robust OCR across scales and transformations is a challenging problem and therefore Google's Tesseract-OCR platform is used as the backend via PyTesseract. The full OCR pipeline is detailed below with a brief explanation for each step.

OCR Processing Pipeline

Input Image	Locate Face and Crop Image	Detect Blobs	Evaluate Blobs	Number Detection
				64 *78% confidence*
• Image containing full body photo of student holding their corresponding number	• In the majority of the images targets are holding the paper such that the number is in line with their face and therefore to reduce search space have cropped the edges of the image based on the face detection • Image crop factor is adjusted for whether it is a frontal picture or a profile image and if no face is found then no cropping is performed	• Image is blurred using Gaussian blurring and then blobs are detected (yellow circles above) based on following parameters: - Minimum blob size - Minimum inertia ratio - Minimum convexity	• Blobs are evaluated to determine whether they are surrounded by a white border (after thresholding image) by calculating the average border color and determining if it is greater than 230 • Remaining blobs are then checked to see whether they are within 30 pixels of one another in order to remove blob duplication from double digit numbers	• Image is converted to grayscale, thresholded, and then sharpened • PyTesseract 3.05 is used on resulting image and is constrained using the command line syntax to only detecting digits • If only one number is to be detected then it takes the argmax of the predicted numbers otherwise it returns all predicted numbers where multiple numbers are found

While this approach did not undergo as robust testing as the facial recognition component, it achieved very high performance on most images when being used for generating folders in the preprocessing stage. It performs nearly perfectly on frontal images although can have challenges with images that are held too sideways for several reasons including difficulty cropping the image and difficulty ensuring that the border is adequately white.

Correctly Predicted **Incorrectly Predicted**

Waldo: 4 63 16 338 5

4.1 Multiple Number Detection

Waldo is not optimized for detecting multiple numbers in images although this functionality is in beta testing. In order to accommodate multiple numbers located in images, the images are not cropped as there are presumably multiple faces located in the same image. Also, rather than taking the argmax of the number predictions, all values are returned. The combination of these steps results in a surprising deterioration of performance as there tend to be many false positives which are predicted.

Example Image with Two Numbers

Waldo Prediction: 64, 65 and 6

4.2 Unsuccessful Approaches

Optical character recognition was far more challenging than originally anticipated and many approaches were attempted most of which involved trying to find the number:

- Detecting the paper was originally thought to be the obvious approach to reducing the search space in detecting the number however this proved far more difficult than expected.
    - Multiple thresholding approaches were applied although failed to overcome two issues: the high variability between images resulted in instability, the white wall posed significant issues in detecting the edges of thresholded images where the target was facing left or right.
    - Edge detection did not perform consistently for similar reasons.
    - Color detection failed as the lighting is inconsistent between cameras and the paper is flexed in certain photos which creates shadows.
    - K-means was also attempted although it did not perform consistently.
    - All of the approaches would work on a subset of the images although required too much manual intervention to be reliable in application. No combination of the approaches was found to be viable.
- Detecting the number was also challenging with PyTesseract requiring a significant amount of tweaking. Multiple versions and implementations were tested including different settings and hyperparameters. Preprocessing the image in conjunction with whitelisting only certain predictions resulted in the most consistent improvement in performance with sharpening the image allowing PyTesseract to detect edges more easily.

4.3 Failure Cases

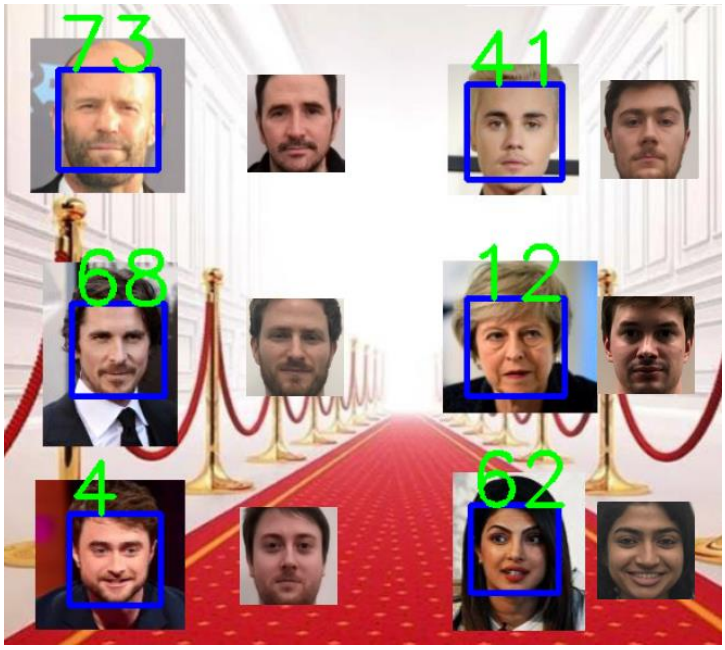
Waldo's largest vulnerability is issues with cropping the picture that lead to an inability to find the number. The first incorrect prediction above is the result of Waldo being unable to detect the face and the second is a result of the body position of the target. The probabilities associated with the predictions output by

PyTesseract are not very intuitive in that certain numbers are expected to be very recognizable and it occasionally has very high confidence in incorrect predictions. PyTesseract is also very sensitive to the scale of the images therefore if an image is too small or large it is unlikely the prediction will be correct.

5 City Goes to the Red Carpet

In order to determine how Waldo performs on faces which are not in the training set I used celebrity faces to find their corresponding match using the logistic regression classifier with HOG features. It is quite evident that one of the main facial features the model uses is the nose and mouth in making its predictions.

Waldo Prediction Based on Celebrity Face Inputs



As facial recognition technology is increasingly being used in many applications it is important to consider the ethical implications as well as the types of performance guarantees these models can make. Research on adversarial attacks suggests CNNs (and other models) can be vulnerable to making incorrect predictions with the addition of random noise which is imperceptible to the human eye (Szegedy et al., 2013). Therefore, it is important to consider best practices in training facial recognition models. As Waldo is biased towards selecting a face from the training set, Theresa May would be able to impersonate Niall on an exam which applies this model to recognize the identity of the test taker.

6 Strengths & Weaknesses of Implementation

6.1 Strengths

- Strong prediction performance on holdout test set including on low-quality images
- Strong OCR performance on most images
- Scalable pipeline with no required manual data preprocessing
- Prediction speed is fast as models are all saved and pretrained
- Able to predict emotions of target and handle multiple number detection

6.2 Weaknesses

- OCR pipeline is too slow to be performed in real-time and relies on a third-party library (Tesseract-OCR)
- Risk of overfitting due to minimal variety in training data even with augmentation
- Model is sensitive to image resolution with images that deviate significantly from the training set posing challenges
- Face detection has room for improvement

7 Future Works and Extensions

- Applying Deep Learning to Face Detection: one of the largest areas for improvement in the detection of faces in group photos which can be significantly improved using state of the art deep learning models such as using R-CNNs (Sun 2018), YOLO (Redmon 2016), and SSD (Liu 2016).
- Augmenting images by adding different rotations, levels of zoom, flipping, and other transformations is a common technique to improve the accuracy of models through their ability to generalize on unseen images (Perez 2017). While many angles of targets faces were extracted, no image augmentation was performed which could increase Waldo's performance.
- Alternative CNN Architectures: VGG16 is no longer state of the art in image classification and alternative architectures including ResNet, Inception, and potentially even AlexNet.

8 Discussion and Concluding Remarks

Waldo is a very accurate facial recognition and OCR application which depends on relatively few external libraries and is easily scalable with an automated data preprocessing pipeline. In developing Waldo, I was surprised by how many online resources there are for "out of the box" facial recognition tutorials in Python using "state of the art deep learning models" and how few high quality tutorials there are for fundamental computer vision concepts like preprocessing images and developing a visual bag of words. In retrospect, this is not surprising given the amount of excitement and new entrants to the field in combination with the large number of prebuilt packages for computer vision tasks including facial recognition although it stands in stark contrast with the lack of depth in support of certain OpenCV functions.

I was very surprised and disappointed by how challenging the OCR component was, particularly in locating the paper. Even with a large number of potential preprocessing approaches, each algorithm only seemed to perform well some of the time as the photos had significant variation.

Waldo's strong performance on photos with low resolution is very impressive and HOG features were far more reliable than I anticipated. I was unsatisfied with ORB's performance on the group test images and am curious to better understand whether the problem lies in the underlying data and the poor resulting descriptors or in an aspect of the implementation.

Given the time constraints I am pleased with Waldo's performance although I would be interested to see the upper limits of its performance by improving the face detection using neural networks, modifying the CNN component to include other architectures, creating an ensemble of approaches, and modifying the OCR pipeline to improve the speed at which the numbers are detected.

REFERENCES

1. Carrier, L., Courville, A. 2013. Challenges in Representation Learning: Facial Expression Recognition Challenge. Dataset available at: https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge/data
2. Dalal, N. and Triggs, B., 2005, June. Histograms of oriented gradients for human detection. In *international Conference on computer vision & Pattern Recognition (CVPR'05)* (Vol. 1, pp. 886-893). IEEE Computer Society.
3. James, G., Witten, D., Hastie, T. and Tibshirani, R., 2013. *An introduction to statistical learning* (Vol. 112, p. 18). New York: springer.
4. Lavrenko, V. 2015. IAML5.9: Gaussian Naive Bayes classifier. Available at: https://www.youtube.com/watch?v=TcAQKPgymLE. Accessed on April 24, 2019
5. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y. and Berg, A.C., 2016, October. Ssd: Single shot multibox detector. In *European conference on computer vision* (pp. 21-37). Springer, Cham.
6. Neurohive 2018. VGG16 – Convolutional Network for Classification and Detection. Available at: https://neurohive.io/en/popular-networks/vgg16/. Accessed on April 24, 2019
7. Perez, L. and Wang, J., 2017. The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621*.
8. Redmon, J., Divvala, S., Girshick, R. and Farhadi, A., 2016. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 779-788).
9. Rublee, E., Rabaud, V., Konolige, K. and Bradski, G.R., 2011, November. ORB: An efficient alternative to SIFT or SURF. In *ICCV* (Vol. 11, No. 1, p. 2).
10. Sun, X., Wu, P. and Hoi, S.C., 2018. Face detection using deep learning: An improved faster RCNN approach. *Neurocomputing*, *299*, pp.42-50.
11. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I. and Fergus, R., 2013. Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199.
12. Thaware, R. 2018. Real-Time Face Detection and Recognition with SVM and HOG Features. Available at: https://www.eeweb.com/profile/rajeevthaware/articles/real-time-face-detection-and-recognition-with-svm-and-hog-features (accessed April 28, 2019)
13. Viola, P. and Jones, M., 2001. Rapid object detection using a boosted cascade of simple features. *CVPR (1)*, *1*, pp.511-518.