Delft University of Technology, Bachelor Seminar of Computer Science and Engineeringa

HaarCNN: Detection and Recognition of Comic Strip Characters with Deep Learning and Cascade Classifiers

Bartlomiej Kotlicki
under supervision of
Lydia Y. Chen, Zilong Zhao
TU Delft

Abstract

Object detection and recognition is a computer vision problem tackled with techniques such as convolutional neural networks or cascade classifiers. This paper tackles the challenge of using the similar methods in the realm of comics strips characters. We approached the idea of combining cascade classifiers with various convolutional neural network architectures for character detection and recognition in consideration of maintaining low computational overhead. The alternative with the selective search algorithm step was also explored. The name of the pipeline is HaarCNN. We compared it to standard methods to verify a potential improvement. We evaluate 750 number of images extracted from comic strips and achieve over 85% precision and around 80% recall of detected faces and over 80% of correct main character recognitions. The images were processed in around 200 seconds. The potentially satisfying results of character annotation can be advantageous in deep learning sub-fields such as generative adversarial networks.

Keywords: deep learning, convolutional neural networks, cascade classifiers, character detection, character recognition, comic strips

1 Introduction

Facial detection and recognition algorithms have become a significant portion of the contributions of the computer vision field to the modern world in recent years. It includes comma.ai's driver monitoring system[1], Microsoft's Windows 10 authentication system, or the automated PARAFE system used at French airports[2]. However, the idea evolves around real-life faces. It doesn't necessarily fit the domain of characters present in cartoon drawings. Why is that? It results from the wide range of different drawing styles - both within the same cartoon and between different ones. The cartoonist draws the same character based on what the character does or where it looks. It means that facial features may significantly differ and not sustain the similarities. Therefore, it is challenging to tune the algorithms to work in this domain.

There are two sides to the coin in regards to deep learning techniques of face detection. On the one hand, they can be precise but computationally intensive. The example which portrays it is Multi-task Cascaded Convolutional Networks[3]. It consists of three separate neural networks: P-Net, R-Net, and O-Net. Training the whole infrastructure to perform face detection is resource-intensive. On the other hand, there are techniques capable of processing images extremely rapidly but prone to higher error rates due to the high variance of faces (or objects to be detected) in unconstrained settings[4]. A known example is Haar feature-based cascade classifiers[5] used commonly in OpenCV open-source library[6].

There are also major differences in approaches towards object detection only using convolutional neural networks. One of the classic approaches is using the sliding window technique on the pyramid of scaled images along with Non-Maximum Suppression. It is a very inefficient approach as a neural network is used to evaluate every portion of the image in different scales but can be very precise. The other approach is using Selective Search algorithm, which allows to return regions of interests for further convolutional neural networks processing. It is significantly faster than the previous mentioned approach.

The challenge of executing face detection on cartoon characters seems exceptionally interesting. The detection of cartoon characters' faces needs to result in a high rate of true positives and a low rate of false positives. However, this task should be achievable without heavy computational workload, preferably without using multiple neural networks, each requiring enormous data sets for training purposes. Character recognition, or automated character labeling, requires effective character detection. The classification of characters for annotation purposes can be used by illustration synthesizers such as conditional generative adversarial networks.

We addressed the feasibility of using the techniques mentioned above in the detection of cartoon characters and the possibility of combining them to achieve low resource intensivity and high accuracy, specifically Haar classifiers and convolutional neural networks. We specifically considered characters present in Dilbert comic strips series. We evaluated precision and recall of characters detected, F1-Score and inference time (the length of processing the set of images).

We showed that it is possible to achieve high recall of characters present in the comic strips using the hybrid approach of providing candidate windows with the Haar feature-based cascade classifier and doing conclusive classification with the convolutional neural network. The computational overhead in this approach was very low. We did it with only a fraction of time compared to the base line of classic approach of of using image pyramid, sliding window and Non-Maximum Suppression technique. Moreover, we show that it was possible to achieve relatively similar or even higher precision or recall by embedding Selective Search step along with Non-Maximum Suppression within this procedure. The computational overhead in this iteration of the hybrid approach was relatively similar.

2 Related Work

Firstly, we present the research that has been done in the field of rapid object detection using cascade classifiers as it is a focal point of the study conducted in this paper. We continue with the discussion about image recognition using convolutional neural networks and end with the overview of the work done on character detection in cartoons. The latter also explores the idea of using a Selective Search algorithm for providing regions of interest.

2.1 Rapid Object Detection

One of the most common ways to perform rapid object detection, which includes identifying faces, is using a boosted cascade of simple features [5]. The algorithm consists of three main components. Firstly, an *integral image* extracts information from features of the image. Secondly, the AdaBoost algorithm constructs a classifier that uses a small number of features within a sub-window of an image. It is used to avoid evaluating redundant features. Thirdly, there is a concept of cascading which is based on the idea of combining multiple classifiers of background features present in the images into a single strong classifier which allows discarding regions of an image that are unlikely to contain desired features. As a result, this classifier called Haar feature-based cascade classifier minimizes computation time while achieving high detection rates[5].

2.2 Image Recognition using Convolutional Neural Networks

The convolutional neural network is a specialized kind of neural network for processing data that has a known grid-like topology[7] like 2D images. They have recently enjoyed great success in the large-scale image and video recognition[8] such as achieving record-breaking results on a highly challenging data set ImageNet using purely supervised learning[9].

They have been used in the detection of faces as well - the task which also falls into the category of image recognition. One of the most effective architectures involved in this is a Multi-task Cascaded Convolutional Network[3]. It consists of three separate networks. First is responsible for obtaining the candidate windows and their bounding box regression vectors. Second, further rejects a large number of false candidates, performs calibration with bounding box regression, and NMS candidate merge. Third outputs five facial landmarks' positions. The problem with this technique is that it requires a massive amount of data for training purposes as there are three major architectures used within the pipeline. It naturally leads to the issue of long training time.

Multiple different architectures are used for image recognition: VGGNet, ResNet, R-CNN and Inception to name a few. These can be used along with the technique of sliding window and Non-Maximum Suppression to remove overlapping windows returned by the classifier or with Selective Search approach for providing regions of interest. Simplified versions of these architectures can be used as well for image recognition. One of the examples is a smaller VGGNet[10] of which the architecture is described in detail in the later part of the paper as it is a major component of the research process.

2.3 Detection of Cartoon Characters

There are multiple papers and repositories on GitHub of which the focus is primarily on the detection of characters present in cartoons. The researchers at the University of Tokyo and Nanyang Technological University have proposed a method of cartoon characters face detection by extracting such features as skin color, edge, jaw contour, and symmetry[11]. However, this paper is primarily focused on the domain of anime characters. The similar work has been conducted on Simpsons characters. The process is based mainly on identifying clusters of white pixels that look like eyes because the characters' faces do not follow a consistent enough pattern[12]. None of these methods tackle the problem by using deep learning techniques.

However, a paper titled "A Study on Object Detection Method from Manga Images using CNN"[13] explores convolutional neural networks for generating region proposals using Selective Search. Selective Search is a region proposal algorithm used in object detection. It is designed to be fast with a very high recall. It is based on computing hierarchical grouping of similar regions based on color, texture, size, and shape compatibility[14].

The work done in this paper follows a similar approach to what the researchers in this paper have done. We wanted to compare the hybrid of Haar classifier and convolutional neural networks in terms of precision, recall, and inference time to process a set of images. Moreover, we extracted different types of regions of interest: the human heads by extracting skin-like color from images or heads of other characters by extracting colors such as white. Next, we used various convolutional neural networks to evaluate if the specific region represents a character.

3 Methodology

We designed a methodology which is abbreviated as HaarCNN. Our goal was to achieve both high recall and precision of characters detected while maintaining the low computational overhead. It is a comic strips character detection approach. The character recognition step is conducted after an object detection step is performed by HaarCNN. The whole character detection and recognition pipeline is presented in Figure 1.

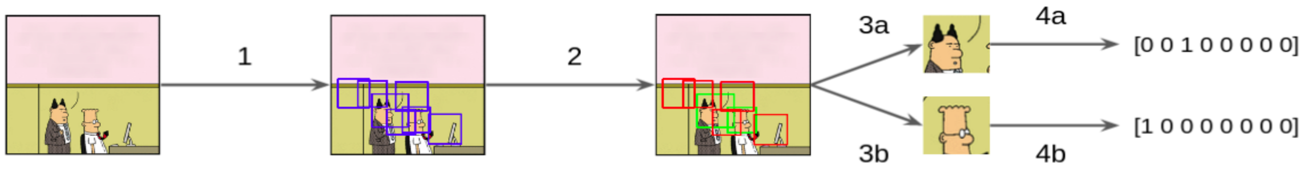

Figure 1: The complete character detection and recognition pipeline. HaarCNN is steps 1 and 2. All steps are described in the following way: 1 - calculate Haar cascade features of the image and return all the windows; 2 - each window is resized to 50 by 50 pixels and passed through a convolutional neural network, if the output is above a threshold then it is considered a face (in the image marked as green window, otherwise red); 3 - each window considered a character is passed to multiclass convolutional neural network intended for classification of a character; 4 - output of the convolutional neural network indicates which character it is.

3.1 Data Preparation

Beforehand, we conduct data preparation for training classifiers used in both detection and recognition. Images of faces present in both comic series and images of non-faces are extracted. All faces are annotated with a number representing a specific character. Optionally, data augmentation is conducted in order to obtain more data for training purposes. Different types of transformations were used such as rotation of an image within the range of 20 degrees, slight change of brightness, horizontal flip or zoom in/out by at most 20%.

3.2 Character Detection

To solve the conundrum of high accuracy in low computational overhead, we proposed two similar pipelines. One focused on combining Haar classifier and convolutional neural networks. The other included a Selective Search step between the Haar and convolutional network classifications. We first give the preliminary of Haar, convolutional neural networks, and Selective Search algorithm, and then explain the two versions of the pipeline to combine all of them.

Classifiers

Different types of classifiers can be used in character detection along with the sliding window technique: Haar feature-based cascade classifier, convolutional neural network along with Non-Maximum Suppression or Selective Search based approach. These can be used separately with varying accuracies and computational times. Potentially, their components can be combined to form a new pipeline.

The Haar classifier is a system that extracts information from the features of the image. There are three types: two-rectangle, three-rectangle, and four-rectangle[5] (see Figure 2). The sum of pixels under white rectangles is subtracted from the sum of pixels under black rectangles. Multiple weak classifiers are constrained by the small number of features with help of the AdaBoost learning algorithm. These construct a single strong classifier that evaluates images based on cascading: instead of applying all features on a window, the features are grouped into different stages of classifiers and applied one by one.

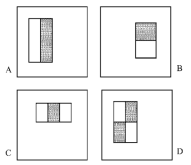

Figure 2: Rectangle features used for training a Haar cascade classifier.

As mentioned in the ”Related Work” section, the convolutional neural network is a specialized kind of neural network for processing data that has a known grid-like topology[7] like 2D images. In this research, we used two types of neural network architectures: a smaller VGGNet (shown in detail in Figure 3) and a simple 3-layer network.

Pipeline of Haar Classifier with Convolutional Neural Networks

The Haar classifier trained on images of objects of interest and objects of no interest quickly returns windows representing these objects due to its low computational intensity. The training itself may be a lengthy process but object detection itself processes images very rapidly[5]. However, it may be prone to higher error rates due to the high variance of objects to be detected[4]. It leads to an abundance of false positives. Potentially, we remedied it by filtering out false positives and retaining as many true positives as possible. We achieved it by applying a convolutional neural network with binary output to windows returned by the Haar classifier. We trained a convolutional neural network with a similar data set: images of objects of interest annotated with ”1” while images of objects of no interest with ”0”.

We discovered two nuances. The Haar classifier’s role was to return candidate windows instead of performing conclusive object detection. We used the convolutional neural network as a conclusive classifier only on these candidate windows instead of on every portion of the image (as in the sliding window technique on a pyramid of scaled images).

Pipeline with Selective Search step

The candidate windows returned by the Haar cascade classifier could be narrowed down by running a Selective Search algorithm on these specific windows. The goal was to look for a particular range of colors within them. These clusters could potentially represent a face of a human-like character or another non-human character like a white dog.

Before the Haar classifier returned candidate windows representing potential character regions, we converted the origi-

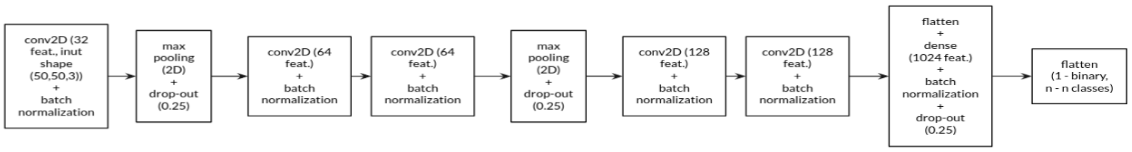
conv2D (32 feat., inut shape (50,50,3)) + batch normalization
max pooling (2D) + drop-out (0.25)
conv2D (64 feat.) + batch normalization
conv2D (64 feat.) + batch normalization
max pooling (2D) + drop-out (0.25)
conv2D (128 feat.) + batch normalization
conv2D (128 feat.) + batch normalization
flatten + dense (1024 feat.) + batch normalization + drop-out (0.25)
flatten (1 - binary, n - n classes)

Figure 3: Smaller VGGNet architecture.

nal RGB image to an HSV image. On this image, we looked for particular RGB ranges of colors: the human-like color of the skin and the white color for a white dog. For each color range, there was a bitwise mask of the image returned. Each displayed clusters of pixels that fall within the specific RGB color range in the HSV image. The Selective Search algorithm was applied on the mask but within the candidate windows to narrow these down. Thus, the Selective Search algorithm is ran only on specific smaller portions of the image instead of on the whole. Afterward, Non-Maximum Suppression was employed to get rid of overlapping windows to retrieve a single one. Every narrowed-down window is then put back on the original image. These depict regions of interest.

On these windows, we applied the conclusive convolutional neural network. In the case of human characters, we used the separate network to distinguish between the face and, for example, an arm of the character. In the case of a white dog, another to distinguish between an actual character and non-character objects like a white shirt or a white cup. The goal was to recognize what illustrated a character and what did not within a specified range of colors.

The character detection portion of the pipeline with detection of human-like characters is presented in Figure 4.

3.3 Character Recognition

Character recognition is a multi-class labeling problem. There was a specific set of characters present to be annotated. In this sub-problem, convolutional neural networks could prove to be effective as well. We needed the convolutional neural network architecture with multiple outputs. We used a smaller VGGNet (Figure 3). The VGGNet was used as it has been proven that it not only achieves the state-of-the-art accuracy on ILSVRC classification and localization tasks but is also applicable to other image recognition data sets, where it achieves excellent performance even when used as a part of relatively simple pipelines[8]. Therefore, it was safe to assume that a simplified version of it would perform well in the domain of drawn characters. The smaller VGGNet led to lower computational time, which was the ultimate goal of the entire research procedure.

Pipeline

Once it was confirmed by the character detection portion of the pipeline that a candidate window showed a character, we passed the image within to the smaller VGGNet with multiple outputs. It verified which character is present in it. We presented this part of the pipeline in steps 4a and 4b in Figure 1.

3.4 Software Architecture

The HaarCNN is explained in a step-by-step manner in Algorithm 1. The alternative pipeline with the Selective Search step is presented in Algorithm 2. The programming language used in the research project was Python.

Algorithm 1: HaarCNN high-level pseudo-code

```
1  threshold ← CNN character detection threshold;
2  face_predictor ← conclusive character detection CNN;
3  char_recognition ← character recognition CNN;
4  faces ← calculate Haar features on an image and
     return candidate windows;
5  for face in faces do
6      face_resized ← resize face to 50x50;
7      output ← pass face_resized through face_predictor;
8      if output >= threshold then
9          character_prediction ← pass pick_resized
             through char_recognition;
10     end
11 end
```

The Selective Search algorithm works in the following way: from set of regions, two are chosen that are most similar in terms of color similarity and texture similarity, these

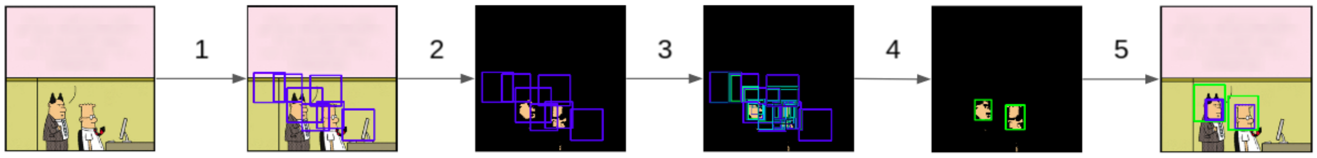

Figure 4: HaarCNN with Selective Search algorithm step. 1 - calculate Haar cascade features of the image and return all the windows; 2 - return bitwise mask of the original image with clusters of pixels presenting human-like characters; 3 - perform Selective Search on candidate windows; 4 - perform non-maximum suppression of boxes returned by the Selective Search algorithm; 5 - put returned windows on the original image. The character recognition portion of the pipeline remains the same as in steps 3 and 4 in Figure 1.

Algorithm 2: HaarCNN (with Selective Search step for human characters only) high-level pseudo-code

```
1  threshold ← certain threshold above which image
     portion after being passed through CNN is
     considered a human face;
2  human_clusters ← bitwise mask presenting pixels
     within RGB range of human skin colour on HSV
     version of original image;
3  human_predictor ← human-like face prediction CNN;
4  char_recognition ← character recognition CNN;
5  faces ← calculate Haar features on an image and
     return candidate windows;
6  for face in faces do
7      rectangles ← perform Selective Search on
         human_clusters but on face window coordinates;
8      picks ← perform NMS on rectangles and return
         non-overlapping windows presenting faces;
9      for pick in picks do
10         pick_resized ← resize pick to 50x50;
11         output ← pass pick_resized through
             human_predictor;
12         if output >= threshold then
13             character_prediction ← pass pick_resized
                 through char_recognition;
14         end
15     end
16 end
```

are combined into a single region and the process is being repeated for multiple iterations. The Non-Maximum Suppression has proposal boxes B, confidence scores S and overlap threshold N as input. The output is filtered proposal boxes F. The pipeline is following: remove the box from B to with the highest confidence score in S (the confidence score could be confidence given by the output of the neural network or normalized area of the box), compare this proposal with other proposals in B, if intersection over union between these box proposals is greater than N and score of other proposal in S is higher, then discard this proposal, otherwise add it to F. These steps are being repeated until all proposal boxes in B are processed. In order to check more details about both, the Selective Search algorithm is explained in [15] and Non-Maximum Suppression in [16].

3.5 Training Procedure

Character Detection

We trained The Haar classifier using the OpenCV command-line application on 2700 images total: 900 representing faces of characters and 1800 presenting objects outside of the face region in the comic strips. We manually extracted the portion of images with the OpenCV interface for Python and performed data augmentation to collect these 2700 images.

There were multiple types of training data used for training convolutional neural networks. All were trained with help of TensorFlow open source library in Google Colaboratory environment. In the main version of the pipeline, we trained the convolutional neural networks (smaller VGGNet and simple 3-layer) to detect objects in the same manner as the Haar classifier on images of characters' faces (annotated with ”1”) and non-character objects (annotated with ”0”). We extracted the images manually. We performed data augmentation to generate more images. The networks were trained on 5000 images.

In the alternative version of the pipeline, we trained the convolutional neural networks differently. We trained the human-like character detection network on images of human-like faces annotated with ”1”. We annotated images of non-faces but of which the dominant color falls within the RGB range of the skin color with ”0”. Data augmentation was used as well for the generation of more images. We based the training process of convolutional neural networks for other characters on the same principle. We used 12 thousand images to train human-like character detection network and around 7 thousand to train white dog character detection network.

Character Recognition

With help of data augmentation procedure, we trained the convolutional neural network for character recognition on one thousand images of characters present in the comic strip series. We wanted to recognize eight main characters. We annotated these with digits spanning from ”0” to ”7”. The convolutional neural network has eight outputs.

4 Evaluation

4.1 Experimental Setup

Data Sets

For evaluation of character detection and recognition of Dilbert comics, we used two data sets. Both of them consist of 750 images of extracted panels from full comic strips. All of them have been modified in a way that they do not contain text. Each ground truth bounding box has a corresponding character annotation attached to it. There are eight main characters used for annotation. Other characters are annotated as ”8”.

Testing

For testing purposes, the **ground truth bounding boxes** are used to verify whether the characters in comic strips are correctly detected and recognized. The ground truth bounding box represents a portion of an image that is supposed to be a true positive. It is annotated by (x,y) top-left of the box coordinates, width, and height of a box. For evaluation purposes, a certain set of comics are annotated with bounding boxes around the faces of characters. Results of the evaluation based on bounding boxes may vary due to the fact that they are manually marked by the human. Therefore, there is a possibility of a situation in which a classifier may correctly detect a character face region but it might be discarded because the returned box may not overlap well enough with the ground truth bounding box. Thus, it may be required to manually check the results and verify if correct true and false positives were given.

Metrics

Intersection over union is a technique to compare the coordinates and area of the box returned by any classifier and the

Classifier type	Haar cascade classifier*	smaller VGGNet for detection	smaller VGGNet for recognition	simple 3-layer network for face detection
Training time (seconds)	1920*	46	42	42

Table 1: Training times for classifiers used in the research.

ground truth bounding box. The formula is following:

$$\frac{intersection \quad of \quad boxes}{union \quad of \quad boxes} \tag{1}$$

Precision is a metric comparing the number of correctly detected objects (true positives) to the number of all detections (true positives and false positives). The number of correct detection is divided by the number of all detections.

$$\frac{true \quad positives}{true \quad positives + false \quad positives} \tag{2}$$

Recall is a metric comparing the number of correctly detected objects to the number of ground truths. First is divided by second.

$$\frac{true \quad positives}{true \quad positives + false \quad negatives} \tag{3}$$

F1-Score is a metric that indicates how good are both precision and recall. The formula is:

$$2 * \frac{precision * recall}{precision + recall} \tag{4}$$

4.2 Training Times

We show training time for different classifiers in Table 1. The Haar cascade classifier was trained on the local machine with Intel(R) Core(TM) i7-6600U CPU @ 2.60GHz. The convolutional neural networks were trained within Google Colaboratory environment utilizing the GPU runtime. Therefore, it is hard to compare training time of Haar classifier with training time of convolutional nets. The training times exclude the time required for data augmentation process.

4.3 Effectiveness of identifying face

How does the main pipeline perform on the extracted panels from comic strips with no text with varying convolutional network threshold? How does it compare to the classic image pyramid, sliding window and Non-Maximum Suppression approach?

To answer the first question, we executed two separate sessions: first with the smaller VGGNet as a part of the pipeline responsible for the detection of a character, and the second with the simple 3-layer network. Both sessions were executed on the first set of 750 panels from Dilbert comics without text. Different thresholds of the convolutional networks were tested for classification. For example, the threshold was 0.5. It means that if the convolutional neural network has given the output of 0.5 or above for character detection within the given window, we accepted it as a character. The range of thresholds used in the experiment is 0, 0.1, 0.2, ..., 0.9, 1.0. The precision, recall, F1-Score, and inference time were measured for each threshold. Ultimately, the returned window was recognized as a true positive if the intersection over the union between the window and ground truth bounding box was 0.4. The reason for that is that candidate windows returned by the Haar classifier could be relatively big compared to ground-truth bounding boxes. Otherwise, it was considered a false positive. The results are presented in Table 2 and Table 3.

The threshold of 0 basically represents how the sole Haar classifier performed without the convolutional network step. It meant that any candidate window provided was considered a character. It led to high recall but very low precision.

To answer the second question, it was required to set up another session in which images from the first set were evaluated on the classifier using the image pyramid, sliding window, and Non-Maximum Suppression. The goal was to ver-

Threshold	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Precision	0.42	0.84	0.84	0.84	0.85	0.86	0.86	0.86	0.86	0.85	0
Recall	0.89	0.81	0.80	0.80	0.80	0.80	0.80	0.80	0.79	0.78	0
F1 score	0.57	0.82	0.81	0.81	0.81	0.82	0.82	0.82	0.82	0.81	0
Inference time (seconds)	248	214	220	205	206	218	220	249	224	201	115

Table 2: The results of running character detection pipeline with the smaller VGGNet. The session is performed on the first set of 750 images of Dilbert comics series without text present in them. Precision, recall, F1-Score and inference time are all measured.

Threshold	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Precision	0.40	0.40	0.41	0.45	0.55	0.68	0.80	0.87	0.92	0.98	0
Recall	0.89	0.89	0.89	0.89	0.87	0.84	0.76	0.64	0.41	0.03	0
F1 score	0.55	0.55	0.56	0.6	0.67	0.75	0.78	0.74	0.57	0.06	0
Inference time (seconds)	241	241	238	236	264	197	178	163	141	104	115

Table 3: The results of running character detection pipeline with the simple 3-layer network. The session is performed on the first set of 750 images of Dilbert comics series without text present in them. Precision, recall, F1-Score and inference time are all measured.

ify what precision, recall, F1-Score, and inference time was obtained with varying step sizes of the sliding window and rescale factor of the images for the image pyramid. However, instead of running it on 750 images, it is executed only on 100. The results are presented in Table 4.

step size / pyramid scale	16 / 1.5	16 / 1.1	12 / 1.1	8 / 1.1
Precision	0.59	0.62	0.60	0.60
Recall	0.72	0.76	0.80	0.80
F1 score	0.65	0.68	0.69	0.69
Inference time (seconds)	2351	2929	4777	10457

Table 4: The results of using image pyramid, sliding window and NMS technique for character detection. The architecture of the convolutional neural network used for conclusive classification is smaller VGGNet. Precision, recall, F1-Score and inference time are all measured.

Firstly, the F1-Score had the similar value over the span of thresholds from 0.1 to 0.9 when smaller VGGNet was used. It was safe to assume that the threshold of around 0.6 was the optimal choice of threshold to be used. In the first session with smaller VGGNet, the values of 0.86 precision and 0.80 recall are relatively satisfying results because the precision oscillates between 0.84 and 0.86 while recall is within the range of between 0.78 to 0.81. Secondly, the inference times differed for the two networks. It is not surprising as a smaller VGGNet is a more complex network than a simple 3-layer network. It resulted in a slightly longer processing time.

Thirdly, in another session the smaller the step size and the rescale factor, the higher the recall but not necessarily the precision. The F1-Score neither improved nor regressed significantly enough. However, it is not the most important observation. The most striking one is the fact that the inference time was substantially higher. It took at least ten times as much time to process 100 images using this technique as to process 750 images with the main pipeline. The HaarCNN approach was superior to this approach by a substantial margin in terms of computational overhead.

How does the main pipeline with Selective Search step perform on the extracted panels from comic strips? How does it compare to the main pipeline with no Selective Search step and the pure Selective Search approach?

To answer these questions, there were four steps taken. The first step verified how the main pipeline performed. We wanted to see how many true positives and false positives are detected compared to ground truths. The threshold chosen for detection was 0.6. The second step was to see how the main pipeline with Selective Search step performs only by detecting human characters. The third step was to verify if adding more characters of which dominant color falls within a different RGB range improved the performance of the pipeline. The new character we wanted to detect was a white dog. The fourth step was to run a Selective Search algorithm for human-characters only on the whole image instead of candidate windows, similar to the approach in [13]. We compared the result of this step to the results of the second step. The results are presented in Table 5. The first set of images of 750 Dilbert comics was used for this experiment.

For first three steps, the results were relatively similar. Still, there was an interesting trend that we noticed. Once we added a new color range, we detected more characters. However, there was a risk of picking up more false positives. Moreover, more neural networks needed to be used. It meant that inference time increased due to more convolutional neural networks doing the data processing work.

For the fourth step, we had to compare the output of Selective Search approach to the HaarCNN with Selective Search step for humans. Looking at Table 5, it is clear that the HaarCNN with Selective Search step was faster in terms of inference time than the object detection based on Selective Search algorithm on the whole image by over 500 seconds. Other metrics were relatively similar.

4.4 Effectiveness of character recognition

How does the smaller VGGNet perform in regards of recognizing 8 main characters in Dilbert comic strip series?

To perform this experiment, two sets of 750 images were used. Different thresholds of the convolutional neural network for face detection were used in the same way as in the first experiment. We verified what portion of main characters (annotated with numbers ranging from 0 to 7) marked with ground truth bounding boxes were correctly recognized.

pipeline version	**HaarCNN**	**HaarCNN + Selective step [H]**	**HaarCNN + Selective step [HD]**	**Selective Search**
true positives	1118	1037	1108	1035
false positive	178	92	158	120
ground truths	1394	1394	1394	1394
precision	0.86	0.92	0.88	0.89
recall	0.80	0.74	0.83	0.74
f1 score	0.82	0.82	0.83	0.81
time elapsed	199	200	221	755

Table 5: The results of running three iterations of the main pipeline and the pure Selective Search approach for extracting human-like character regions of interest in the whole image. The HaarCNN column corresponds to running a main pipeline of Haar classifier returning candidate windows and convolutional neural networks for conclusive classification. The next two columns correspond to expanded version of the main HaarCNN pipeline with Selective Search algorithm added between Haar and convolutional neural networks classifiers. [H] stands for human-like characters detection and [HD] for both human-like characters and a white dog. The data set used is 750 images of Dilbert comics with no text. True positives, false positives, ground truths, precision, recall, F1-Score and inference time are all measured.

The actual numbers of character recognitions could be lower due to the fact that face detection portion of the pipeline still picked up false positives. The results are present in Table 6.

Threshold	0	0.2	0.4	0.6	0.8
Character recognitions in set of images 1	0.86	0.88	0.88	0.88	0.88
Character recognitions in set of images 2	0.85	0.86	0.86	0.87	0.87

Table 6: The results of using smaller VGGNet for character recognition with different thresholds of the convolutional neural network. We verified what portion of eight main characters marked with ground truth bounding boxes were correctly recognized. Two sets of 750 images of Dilbert comics with no text were used.

Looking at these results, it is clear that the smaller VGGNet architecture performed well on the recognition of characters. At least 85 percent were correctly identified. There are certain probable reasons why it was not higher. False positives were picked up from the face detection portion of the pipeline. It meant that the network tried to recognize the character in a situation where there was no character present. It also explains why the higher the threshold of the face detection network, the higher the ratio of correctly identified characters. The higher threshold led to higher precision which meant fewer false positives. In this experiment, it translated to fewer misclassifications.

5 Responsible Research

5.1 Ethics

When it comes to the usage of comics strip images, the data that has been used in the research has been acquired from *dillbert.com*. We have used the comics for purely educational purposes without any intention of profiting from the work of the creators of this particular comic strip series.

The desire to decrease the computational intensity was partially inspired by the fact that a lot of work being done in the field of deep learning is resource intensive. This is due, for example, to the usage of graphics processing units for performing complex calculations which is energy intensive and not environmentally friendly. This problem was an inspiration to perform object detection and recognition in an efficient manner.

5.2 Reproducibility

This research project is reproducible. The interested entity needs to follow the steps in the section titled ”Methodology” - specifically the subsection titled ”Software Architecture” in which the higher level pseudo-code has been described for both alternatives of the pipeline.

6 Conclusions and Future Work

The two alternatives of the HaarCNN pipeline for detection of comic strip characters displayed three common key features. Firstly, they were fast. The rapid object detection technique using a Haar cascade classifier for returning candidate windows in combination with conclusive classification accomplished with the convolutional neural networks led to satisfactory inference times compared to other baselines. The same applied to the alternative approach with Selective Search step. Secondly, both alternatives gave high precision and recall. The recall of characters present in the comic strips series could be especially higher when the alternative with a Selective Search step was used for more types of characters. Thirdly, given the solid precision and recall ratios the character recognition could lead to exceptional results which can prove to be useful for automated annotation in deep learning domains such as illustration synthesizers with generative adversarial networks for the purpose of automated image annotation. However, there are a couple of things which also need to be addressed. The whole pipeline is going to work only on the very specific comics strip series. It is domain specific. It is impossible to use techniques verified and trained for a specific comic strips series in this research and apply them to a different domain of drawn characters.

References

[1] “Frequently asked questions - commapilot.” https://comma.ai/faq.

[2] “Parafe: a new generation of smart gates for the adp group.” https://www.thalesgroup.com/en/markets/digital-identity-and-security/government/customer-cases/smart-gates-paris.

[3] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, “Joint face detection and alignment using multitask cascaded convolutional networks,” *IEEE Signal Processing Letters*, vol. 23, no. 10, pp. 1499–1503, 2016.

[4] B. Yang, J. Yan, Z. Lei, and S. Z. Li, “Aggregate channel features for multi-view face detection,” in *IEEE international joint conference on biometrics*, pp. 1–8, IEEE, 2014.

[5] P. Viola and M. Jones, “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 2001*, vol. 1, pp. I–I, IEEE, 2001.

[6] “Cascade classifier training.” https://docs.opencv.org/3.4/dc/d88/tutorial_traincascade.html.

[7] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[8] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[10] A. Rosebrock, “Keras and convolutional neural networks (cnns).” https://www.pyimagesearch.com/2018/04/16/keras-and-convolutional-neural-networks-cnns/.

[11] K. Takayama, H. Johan, and T. Nishita, “Face detection and face recognition of cartoon characters using feature extraction,” in *Image, Electronics and Visual Computing Workshop*, p. 48, 2012.

[12] K. Tokis, “Kostastok/simpsons-face-detector.” https://github.com/KostasTok/Simpsons-Face-Detector.

[13] H. Yanagisawa, T. Yamashita, and H. Watanabe, “A study on object detection method from manga images using cnn,” in *2018 International Workshop on Advanced Image Technology (IWAIT)*, pp. 1–4, IEEE, 2018.

[14] V. S. Chandel, “Selective search for object detection (c++ / python).” https://learnopencv.com/selective-search-for-object-detection-cpp-python/.

[15] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders, “Selective search for object recognition,” *International journal of computer vision*, vol. 104, no. 2, pp. 154–171, 2013.

[16] S. K., “Non-maximum suppression (nms).” https://towardsdatascience.com/non-maximum-suppression-nms-93ce178e177c.