AN OBJECT DETECTION AND ITS EDUCATIONAL EFFECT ON ASPIRING FILM DIRECTORS USING COMPUTER VISION AND MACHINE LEARNING

Fengrong Han[1] and Ang Li[2]

[1]Veritas Collegiate Academy VA, 5225 Backlick Road Springfield，VA 22151
[2]California State University, Long Beach,
1250 Bellflower Blvd, Long Beach, CA 90840

ABSTRACT

The issue that is being addressed is how those who wish to be involved in film production in the future can be more educated regarding what goes into making a movie. An application was created to help these people keep track of movie props and characters and where they are placed throughout a movie. This is performed by using videos or images as input for the application, and the application uses an object detection model from MediaPipe to determine what types of objects are in a scene and where exactly they are located in the scene [4]. Then, bounding boxes are placed around the objects with the classification of the object next to each bounding box, and the result is returned. The results of the experiment indicate that the application is consistent at detecting objects when the object is fully shown on the screen. However, when the application used the cropped images as input as opposed to the full version of the image, the application was not consistently accurate, as it was not able to recognize objects that were partially cut out of the picture. More work will be needed to improve both the object detection capabilities of partially obstructed objects and the overall user interface of the application [5].

KEYWORDS

Movies, Object Detection, MediaPipe, Application.

1. INTRODUCTION

Object detection is a topic that has been gaining traction in recent years [6]. With advancements in technology, object detection is becoming more commonplace and accessible for the general public to use [7]. Object detection models can be fed images as input, then return the same images back with a bounding box around each object they identify as well as what each detected object was classified as [8]. Such a technology could open up the potential for new tools to be created and bring convenience to people. However, some may view the advancement of object detection in a negative light, as cameras that are set up in public with facial detection capabilities may make some feel uncomfortable and insecure about their privacy.

Object detection is a significant topic today because it is a tool that can have applications across many different fields [9]. For example, in the medical field, having specialized object detection models could detect abnormalities in X-rays or CT scans and accurately perform an early diagnosis of an issue to prevent the worsening of the issue in the future. When it comes to film

David C. Wyld et al. (Eds): MLDS, IBCOM, CSIA, NLAI - 2022
pp. 25-34, 2022. CS & IT - CSCP 2022 DOI: 10.5121/csit.2022.122103

students, there is much information to be gained from watching a movie and picking apart the various elements and intricacies that make up the movie as a whole. Something that could benefit film students in their education is a tool that could automatically tell them when and where certain movie props or characters are shown in a movie. Therefore, object detection can have massive benefits to our society.

One way that the task of counting the number of instances that a movie prop appears in a movie could be performed is by watching the entire movie and manually keeping track of how many times a specific movie prop shows up on the screen. This could be done on paper or recorded on an electronic device. However, this strategy has several major downsides that makes it unreliable and inconsistent in the long run. The person who is keeping track of the movie prop may miss its appearance in some scenes, especially if the movie itself is fast-paced and chaotic. Furthermore, some props may seem similar to each other, and a person who tries this strategy might confuse them and count more instances of a prop's appearance than there actually is. One way of countering such an issue is by getting a sample of participants to watch the movie in its entirety and count how many times the prop appears. Then, the average of all the answers rounded to the nearest whole number can be used as the final count. While this solution may yield more accurate results, the process itself has much room for human error and takes too much time and effort to be worthwhile as a long-term solution. Another possible method is asking someone within the movie production team directly regarding how many scenes a movie prop was used in a scene. However, this method will require having relations with a member of the movie production team, which the majority of the general public will not have. With how busy the members of the production team may be, it is also not guaranteed that the answer will be given in a timely manner. Therefore, such a strategy will not be applicable to most people. Currently, there is no quick, reliable way that one who is unacquainted with a member of the movie production team can easily determine the number of times a specific movie prop appears in a movie.

A tool that can be used to help film students in their goal of becoming art directors is an application that keeps track of movie props. To do so, the application can be fed images or videos as input and can classify what props are present in the images or videos and where exactly they are located. By using an object detection model in the application, the application provides bounding boxes around each object and states the object type next to each bounding box [10]. Compared to the method of someone manually going through a movie and counting how many times a movie prop shows up or what scenes the prop shows up in, the application can be much quicker at coming up with a result and potentially more accurate as well. The main strength with the application is that the process of movie prop detection is more streamlined and convenient. A user of the application can simply input an image or video and come back to view the results once the object detection model has finished. While individual images may be easier to determine manually, inputting videos could allow the user to input the video, wait for the application to finish loading, and come back to view the results.

To prove the effectiveness of the application at accurately identifying props in movies, thirty pictures were used as a sample size for the application to analyze. The pictures are selected so that they are as different from each other as possible, which will provide more variety for the application input. With each one of the pictures, ten different cropped portions of the picture will be used as the input. The results of how many times the application successfully identified the objects of each image with different crops will then be recorded in a table. The goal of this experiment is to see how well the application can work when presented in various settings and environments to see if the application works much better in certain settings than others. For example, one of the images could be taken underwater, while another image could be taken in the mountains. If the application is only fed very similar images, it will be very difficult to accurately determine the performance of the application among all types of images. Furthermore, different

crops of the image means that the application can also be tested for whether it can detect objects that are partially cut off from the image. Object detection models generally do well when they are provided with an image that shows an object in its entirety in front of a clean background. However, in movies, this will not happen all the time, and some objects will only be partially shown in some scenes. Therefore, it is important that the application is able to detect the objects in these scenarios.

The remainder of the paper is structured into five other sections. The next section, Section 2, offers a look into the challenges that were encountered when creating the experiments and developing the application. Section 3 provides an insight on how the application was made with both a general explanation and in-depth explanations of certain aspects of the application. Section 4 describes the experimentation process of the application. Section 5 compares and contrasts related works to this work. Lastly, Section 6 summarizes the project and experiments in a conclusion, and self-reflections and plans for the future of the application are mentioned as well.

2. CHALLENGES

In order to build the project, a few challenges have been identified as follows.

2.1. Choosing the features

The first major challenge when starting the project was choosing the features that would be included in the application. The application was meant to help aspiring movie directors learn the movie-making process and what is needed to make a movie [11]. However, too much information would be needed for film students to fully grasp the idea of what it takes to make a movie. Therefore, the specific feature that was chosen to be focused on was movie props. Almost every movie uses props of some sort to create plot devices or settings and environments. In terms of movie props, they would generally be reused across multiple scenes or in different shots. One way to express this idea to aspiring film directors is to create an object detection model that can pinpoint exactly where a prop is used in each scene; this would hopefully help them envision how they could use props in their future movies.

2.2. Deciding what technologies to use

The next challenge was deciding what technologies to use when building the application. With object detection as a core feature of the application, finding the right object detection model that was accurate, easy to implement, and quick to compute was crucial [12]. After some searching, MediaPipe seemed to be the most promising option. MediaPipe has 3D object detection capabilities, which could allow the detection of movie props. MediaPipe is particularly specialized in its ability to recognize parts of the human body, such as hand tracking and face tracking. This would make MediaPipe ideal for being able to detect not only props that appear in the movie but also characters that appear in the movie. Since MediaPipe is mainly implemented in Python, the application would need to use an API to connect the Python code and the user interface code together [13]. To achieve this, Flask was used as the web framework, and the implementation of Flask was fairly simple.

2.3. Coming up with an experiment to test the application with

The final challenge was coming up with an experiment to test the application with. When released to the general public, the application would work with a variety of scenes to classify objects from, and the goal was to ensure that the application would be robust enough to

accurately detect objects in a multitude of filters and environments. To address this, the application was tested with images that were as different from each other as possible. Another potential issue was that movie props in their movies would not always be shown in their entirety. Because of the way some shots are taken, props may be somewhat obstructed or otherwise not clearly shown in a scene of the movie. A way of handling this issue was to take the original images that would be used as input and send ten different cropped version of the image as input. By doing so, objects in the image would only be partially shown and would test the application's ability to detect parts of an object.

3. SOLUTION

This application takes in inputs of images, and returns a list of names of items and the amount of each item that appears in the image.This project contains three major sections: core algorithm, server, and APP UI. The core algorithm is constructed via a pytorch object detection model, yolov5x [15]. It takes in a jpg, jpeg, or png image file and returns a dictionary of names and amount for each item in the image.The second part, the server, is constructed via the flask library of python. It requests an image input, sends it to the core algorithm, and returns the final results.The third part, the APP UI, is constructed via flutter, an UI software created by google. It serves as the front end of the project where the user sends the image to the server for processing.

The project is built on three segments of code, causing the connection between them the hardest technical difficulties. The pretrained model, yolov5x, shows limitations on the range of objects as well as its ability to detect overlapped objects that appear in the image.

Solution:

- A wide range of categories of objects is listed, such as sports gear, stationeries, fruits and vegetables etc.
- Then the model is tested according to the following criterias:
    - Images containing objects from one of the categories
    - Images containing objects from multiple categories
    - Objects without intersecting
    - Objects with some level of intersection
    - Objects largely overlapping

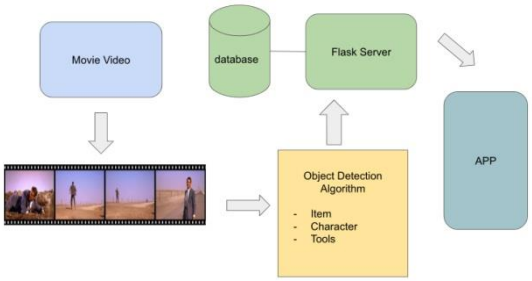
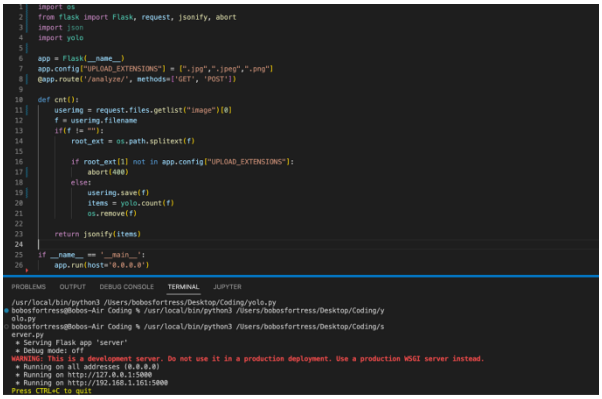


Figure 1. Overview of the solution

```python
import torch
import pandas as pd

def count(img):
    mdl = torch.hub.load("ultralytics/yolov5", "yolov5x", pretrained = True)
    pics = [img]
    res = mdl(pics)

    df = res.pandas().xyxy[0]

    items = {}

    for index, row in df.iterrows():
        if(row["name"] in items.keys()):
            items[row["name"]] += 1
        else:
            items[row["name"]] = 1

    return items
```

Figure 2. Core-Algorithm:

First and foremost, the pretrained machine learning model yolov5 is loaded from the library (Line 5). Yolov5 x is specifically chosen out of the 5 models of yoloc5 library due to its best performance. This model is trained on 1.5 million object instances for 300 epochs using the COCO dataset. We used pandas function from the pytorch model to access the coordinate, name, and amount information from the result variable from yolov5 x (Line 9). At last, a loop that increments the amount of objects under each key (unique item name) in the dictionary is written and implemented into the dictionary variable named items, which is returned as the final result of this algorithm.



```python
import os
from flask import Flask, request, jsonify, abort
import json
import yolo

app = Flask(__name__)
app.config["UPLOAD_EXTENSIONS"] = [".jpg",".jpeg",".png"]
@app.route('/analyze/', methods=['GET', 'POST'])

def cnt():
    userimg = request.files.getlist("image")[0]
    f = userimg.filename
    if(f != ""):
        root_ext = os.path.splitext(f)

        if root_ext[1] not in app.config["UPLOAD_EXTENSIONS"]:
            abort(400)
        else:
            userimg.save(f)
            items = yolo.count(f)
            os.remove(f)

    return jsonify(items)

if __name__ == '__main__':
    app.run(host='0.0.0.0')
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  JUPYTER
/usr/local/bin/python3 /Users/bobosfortress/Desktop/Coding/yolo.py
bobosfortress@Bobos-Air Coding % /usr/local/bin/python3 /Users/bobosfortress/Desktop/Coding/y
olo.py
bobosfortress@Bobos-Air Coding % /usr/local/bin/python3 /Users/bobosfortress/Desktop/Coding/s
erver.py
 * Serving Flask app 'server'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://192.168.1.161:5000
Press CTRL+C to quit
```

Figure 3. Server

Firstly, a specific route, "/analyze/", is created for the flask server, where the core algorithm will be hosted (Line 8). When the code is running, a server IP address is generated, where the file would need to be uploaded for successful access. Next, an image is requested from the user (Line 12). After checking the nullity of the image (Line 14) and if file extension fits the specified types (Line 17), 'jpg, jpeg, or png' (Line 7), the image is passed to the core-algorithm, yolo.count. The returned results are saved as a dictionary which is then returned in a json format.

```
24      void getImage(ImageSource source) async {
25        PickedFile? file = await smart.getImage(source: source);
26
27        setState(() {
28          selectedImage = File(file!.path);
```

Figure 4. APP UI

This is the main function of the image picker page. It allows the user to access images through the image gallery of their operating system to upload to the server.

```
class _MyNextPageState extends State<ResultPage> {
  Map<String, dynamic> items = {};
  @override
  void initState(){
    super.initState();
    uploadFile();
  }
  void uploadFile() async{
    var url = "http://127.0.0.1:5000";
    Map <String, String> serverInfo = {"Connection": "Keep-Alive", "Keep-Alive": "timeout=5, max=1000"};
    print("$url/analyze/");
    http.MultipartRequest request = http.MultipartRequest("POST", Uri.parse("http://10.0.2.2:5000/analyze/"));
    request.headers.addAll(serverInfo);
    request.files.add(await http.MultipartFile.fromPath("image", widget.imageFile.path, contentType: MediaType("application", "jpg")));
    request.send().then((r) async {
      print("statusCode: ${r.statusCode}");
      if(r.statusCode == 200){
        var result = await r.stream.transform(utf8.decoder).join();
        print("result: ${result}");
        setState(() {
          items = jsonDecode(result);
          print("items: ${items}");
        });
      }
    });
  }
```

Figure 5. Core segment

This is the core segment of the result page.

The IP address generated by the flask server is implemented (Line 26 & 29), which connects these two parts of the project. The connection is sustained for a period of time (Line 29), where the image file picked on the second page would be posted onto the server during the connection. In the end, the result is printed on the UI in a viewable format.

User access U, pick image from gallery, send it to S, S requests the img, process via C, C returns the results to S, S then posts the results to U which then is printed in modified formatting.

4. EXPERIMENT

4.1. Experiment 1

The experiment that will be used to test the effectiveness of the application at identifying and location props in a movie is the carefully selected inputs of thirty different images. The images are selected based on how different they are from each other in terms of the background and setting as well as the object itself that is to be detected. For example, one image features a brightly colored ball in a grassy field, while another image features a house located in a mountainous region. To reduce any confounding factors, the images that were chosen. With each of these images, ten different cropped variations of the images are produced. The first variation will be the unchanged image, but the nine other variations are consistently cropped to show the top-left portion, the top-middle portion, the top-right portion, the middle-left portion, the middle portion, the middle-right portion, the bottom-left portion, the bottom-middle portion, and the bottom-right portion. Each picture is treated equally in this regard in hopes that confounding variables are prevented.

Image #	Object Detection Accuracy	Accuracy Percentage
1	10/10	100%
2	9/10	90%
3	9/10	90%
4	10/10	100%
5	8/10	80%
6	7/10	70%
7	10/10	100%
8	10/10	100%
9	8/10	80%
10	7/10	70%
11	10/10	100%
12	9/10	90%
13	10/10	100%
14	6/10	60%
15	8/10	80%
16	10/10	100%
17	9/10	90%
18	9/10	90%
19	6/10	60%
20	5/10	50%
21	10/10	100%
22	8/10	80%
23	10/10	100%
24	3/10	30%
25	9/10	90%
26	8/10	80%
27	10/10	100%
28	9/10	90%
29	9/10	90%
30	6/10	60%

Figure 6. Table of accuracy

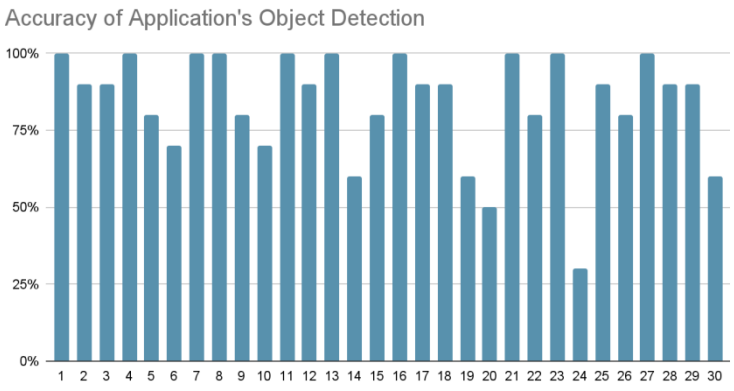


Figure 7. Accuracy of application's object detection

According to the results of the experiment, the object detection seems to score somewhat highly with the majority of images. Out of the thirty images that were tested, ten of them had a perfect score with all ten cropped variations in terms of being able to accurately detect the object on the screen. The lowest accuracy out of all the images was 30%, and all the other images had an

accuracy of 50% and above. When analyzing which pictures did well and which pictures did not, an overwhelming majority of the cropped variations that were the full images were accurately detected. On the other hand, many of the images that didn't produce accurate results from the video were cropped images that were not the full image. In the variations of the images that had parts of the image cropped out, the object in the image was often partially cropped out as well, which can be an explanation for the disparity of the object detection algorithm's accuracy between uncropped and cropped images.

From performing this experiment and looking through the results, it appears that the object detection performed significantly better when the image fully displayed the image to be detected within it. However, the application did not do as well when the image was cropped so that the object was partially cut off. This is an expected result, as the object detection was likely trained and tested using images with the objects fully in view. On the other hand, images with objects that were obstructed or only partially shown were likely not trained for as extensively in the object detection model. If this application were to be widely released to the general public, the application's object detection feature would work fairly well in most scenarios. However, there is much room to make the object detection even more accurate and consistent at even more scenarios, which can make the application more reliable and trusted in the long run.

5. RELATED WORK

Chu and Guo have performed research on movie genre classification, which is done by using neural networks to find the genres that a movie poster can fit into. By tackling this multi-label image classification problem with a convolutional neural network for object detection, the performance of the model at genre classification improved significantly [1]. The article by Chu and Guo and this work share the similarity of conducting research regarding movies. Chu and Guo uses image classification to differentiate between the different genres of movies. On the other hand, this work involves object detection to detect movie props in an application.

Borse et al. described an object detection application that was designed for a robot. By attaching a camera to a robot, the robot would be able to take in images as input, run the images through an object detection algorithm, and make a decision based on the objects that are detected. For example, the robot could treat the objects as obstacles and navigate around them [2]. Both the research of Borse et al. and this work revolve around object detection. While the related work places a greater emphasis on the implementation of object detection on a robot, the focus of this work lies in testing the ability of the application to perform well at detecting movie props in movies.

Zhao et al. performs a study regarding the integration of deep learning into object detection methods. Traditional object detection architectures are compared to object detection architectures based in deep learning, and the convolutional neural network is a tool from deep learning that has the capability to improve future object detection systems [3]. The work of Chao et al. and this work are both similar in that the main general topic is object detection. However, Zhao et al. go further in depth regarding the details behind the implementation of object detection itself, and this work instead focuses on using object detection in an application and experimenting on said application.

6. CONCLUSIONS

An application was designed primarily to aid film students in their journey to becoming full-fledged movie directors by providing them with a convenient method to determine how and

where movie props are used in movies and potentially provide these students with ideas for how movie props can be applied in their own future movies. To have the application figure out where the movie props are, an object detection algorithm is implemented in the application that allows images to be inputted. The application then outputs the. To test the effectiveness of the application at accurately determining what objects are in a movie scene and where they are located in a scene, the application was provided with thirty images. With each image that was inputted, the image was inputted ten times, and the image was cropped with various dimensions before being inputted each time. The number of times that the application can accurately detect the objects was recorded in a table for each image. According to the results, the application appears to perform fairly well when it comes to classifying and locating objects in various scenes and image crops. However, a noticeable pattern when testing the application is that whenever the image is cropped in a way that only a portion of a movie prop is shown on the screen, the application is much less likely to recognize the existence of the object in the image. Moving forward, hopefully a more advanced object detection algorithm can be created to more accurately detect objects that are only partially shown on the screen. Participants who tested the application generally felt positively about the application and believed it would be a useful tool.

One major limitation to the application is the design of the application. While the application successfully performs object detection, the user interface of the application appears quite bare-bones to the point where users may avoid using the application or giving the application a try in the first place [14]. Many applications on the market have a clean yet professional design, and if the general public recognizes that the user interface pales in comparison to that of other applications, they will most likely be attracted to those other applications instead. Therefore, the user interface of the application is likely the biggest limitation that the application currently has.

Something that can be done is adding a little more color to the application, such as simple patterns for the background or colors to some of the text and buttons. The elements in the user interface could also be spaced more to fill up more of the screen and potentially make the application appear less empty.

REFERENCES

[1] Chu, W.-T., & Guo, H.-J. (2017). Movie genre classification based on poster images with Deep Neural Networks. Proceedings of the Workshop on Multimodal Understanding of Social, Affective and Subjective Attributes, 39–45. https://doi.org/10.1145/3132515.3132516

[2] Borse, H., Dumbare, A., Gaikwad, R., &Lende, N. (2012). Mobile Robot for Object Detection Using Image Processing. Global Journal of Computer Science and Technology Neural & Artificial Intelligence, 12(11).

[3] Zhao, Z.-Q., Zheng, P., Xu, S.-T., & Wu, X. (2019). Object detection with deep learning: A Review. IEEE Transactions on Neural Networks and Learning Systems, 30(11), 3212–3232. https://doi.org/10.1109/tnnls.2018.2876865

[4] Lugaresi, Camillo, et al. "Mediapipe: A framework for perceiving and processing reality." Third Workshop on Computer Vision for AR/VR at IEEE Computer Vision and Pattern Recognition (CVPR). Vol. 2019. 2019.

[5] Reynolds, Dennis A., and Gernot Metze. "Fault detection capabilities of alternating logic." IEEE Transactions on Computers 27.12 (1978): 1093-1098.

[6] Papageorgiou, Constantine, and Tomaso Poggio. "A trainable system for object detection." International journal of computer vision 38.1 (2000): 15-33.

[7] Papageorgiou, Constantine P., Michael Oren, and Tomaso Poggio. "A general framework for object detection." Sixth International Conference on Computer Vision (IEEE Cat. No. 98CH36271). IEEE, 1998.

[8] Gao, Tianshi, Benjamin Packer, and Daphne Koller. "A segmentation-aware object detection model with occlusion handling." CVPR 2011. IEEE Computer Society, 2011.

[9] Hu, Han, et al. "Relation networks for object detection." Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.
[10] Gao, Tianshi, Benjamin Packer, and Daphne Koller. "A segmentation-aware object detection model with occlusion handling." CVPR 2011. IEEE Computer Society, 2011.
[11] Mutlu, Nadide GizemAkgulgil. "The future of film-making: data-driven movie-making techniques." Global Journal of Arts Education 10.2 (2020): 167-174.
[12] Zhao, Zhong-Qiu, et al. "Object detection with deep learning: A review." IEEE transactions on neural networks and learning systems 30.11 (2019): 3212-3232.
[13] Lugaresi, Camillo, et al. "Mediapipe: A framework for building perception pipelines." arXiv preprint arXiv:1906.08172 (2019).
[14] Snilstveit, Birte. "Systematic reviews: from 'bare bones' reviews to policy relevance." Journal of development effectiveness 4.3 (2012): 388-408..
[15] Zhou, Yue, et al. "Mmrotate: A rotated object detection benchmark using pytorch." arXiv preprint arXiv:2204.13317 (2022).