Book Cover Recognition

COMPENG 4TN4: Image Processing - Project 2 Report

Harneet Singh - 400110275

Abstract

With this project, I have developed a methodology using which people can identify the names of the books and also, find out dimensional information regarding the text book. The idea is implemented such that users will turn on the live feed using their cameras and the program while working in the background will recognize the key features in the frame to classify the book correctly. It is important to gather the book features before a certain book is tested, because similar to image recognition, we would not be able to determine the name of the book if it is not in the database. Techniques such as image warping, knn (k-nearest neighbors) matching for classification, and contour parameters are used to precisely measure the dimensions of the book. Some of the challenges such as depth perception are also tackled in this project by utilizing the geometric manipulation and by creating an environment that best identifies the framework. An advantage of this technique is that we can store the features extracted from the database and in the future, we may create eigen faces to reduce the required memory size and it will allow for faster classification of objects in an image/frame. I also compared the book recognition task by implementing Convolutional Neural Network (CNN) and Artifical Neural Network (ANN) networks to analyze the accuracy and loss metrics.

1. Introduction

With online shopping reaching new heights, it is inevitable that most companies would require their products and services to be offered online. As can be witnessed, the usual goal of these companies is not just to bring their products online but also to add an additional feature which allows their users to compare different products on their website and also against their competitor's products. In order to achieve these objectives, it is logical to invest in the development of techniques and algorithms that can quickly and accurately identify the products. These trends are also seen in the realm of book selling market and therefore, it is important that developers create book cover recognition algorithms. The aim of these algorithm should be to classify the books based on genre, author, price, name and other relevant information related to selling points of books. The goal of my project is to classify the images based on the features associated with the book cover. In the future, more work may be performed to create apps and websites that can offer these services online and in real time to better assist the users.

In this project, my program extracts features from images containing books and creates descriptors for the said features. The goal of adding descriptors is to classify the image and draw a bounding box around the book in a real time camera feed (or video). Before finding features in the image, first job would be to create a database that will contain images with their labels. In order to create a strong database, image pre-processing steps will be applied to remove unwanted variations from the data images. After resizing the images in the database and also from the camera feed, we can compare the descriptors between two images using knn matching. I have tested the classification through knn and FLANN matchers. FLANN comprises algorithms that are optimized for fast nearest neighbor search in large datasets and for high dimensional features [5]. In the end, object needs to be localized in each frame of the video and a marker will be attached to it to specify its class. This way object recognition task will be achieved for this project.

Additionally, a stretch goal is implemented to measure the dimensions of the book and add a feature that calculates total pages (approx.) when the book thickness is in the frame. To achieve this functionality, I had to apply filters such as resizing, smoothing filter, edge detection to binarize the image and morphological operators to get the best input image data.

Also, after implementing the object recognition steps, I have attempted to compare the results of my project with pre-existing object recognition classifiers.

1.1. Related Work

At Stanford University, Linfeng Yang and Xinyu Shen created a GUI in MATLAB to determine information about a desired book in real time [7]. After taking pictures of

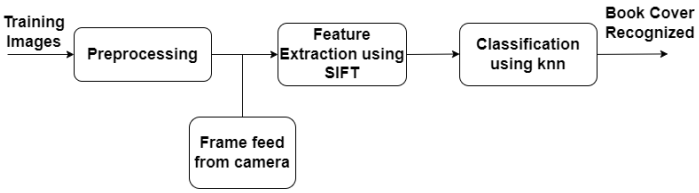

Figure 1. Proposed Method Flowchart

the book cover, the app detected feature of the input image based on Maximally Stable Extremal Region (MSER) algorithm. They captured the textual and non-textual information separately based on morphological difference. To get optimal results, they created a text character alignment algorithm which improves the accuracy of original text detection. Later, they compared their algorithm with MATLAB's built-in OCR algorithm and a commonly used open source OCR. Further, to eliminate false detection inhibition and word correction, they implemented a post detection algorithm and natural language processing. On the online platform, their algorithm attained accuracy numbers of greater than 86%.

In the past, work has been conducted on OCR technology namely, Tesseract OCR engine [6]. OCR engine has been used to perform identification task on natural language. In their experiments, they analyzed the Tesseract OCR along with Transym and their conclusion was that Tessearct OCR is more stable when it comes to large image databases.

Previously, other techniques have been used to extract features from images such as SIFT algorithm. Technically speaking, in my opinion, any work performed to improve the accuracy of object classification in an image task is relevant to the book cover recognition because similar principles can be applied in book cover recognition. The simplicity and efficacy of SIFT algorithm inspired me to work on this project using SIFT algorithm while performing preprocessing steps. The biggest advantage of applying SIFT algorithm is that it can identify the features without regard to orientation and scale (to a certain degree) of the book.

2. Proposed Method with Results

First step is create a database with train images and their labels. After that, preprocessing steps have been applied to acquire consistent images and images are converted to grayscale to reduce the number of channels associated with train images. To extract the features, descriptors and keypoints are stored for each image using SIFT algorithm and subsequently, knn-matcher is used to classify the camera frames accurately. Finally for measuring the book dimensions, preprocessing steps are applied which will be explained in the following sections. Flowchart is shown in Fig. 1.

Figure 2. Collage of Train Images (Original)

Figure 3. Collage of Train Images (Resized and grayscaled)

2.1. Train Images Dataset

To create a dataset, I am utilizing the Harry Potter book collection. Book cover images are found using an online source and labels are attached to each image as the image's name (as shown in Fig. 2).

Labels: ['HP and the Chamber of Secrets', 'HP and the Deathly Hallows', 'HP and the Goblet of Fire', 'HP and the Half-Blood Prince', 'HP and the Order of the Phoenix', "HP and the Philosopher's Stone", 'HP and the Prisoner of Azkaban']

2.2. Resizing and Grayscale

After gathering the images, their sizes are reduced to a scale of 60% (empirically tested, program allows for alteration of scale factor) and images are converted to grayscale (as shown in Fig. 3).

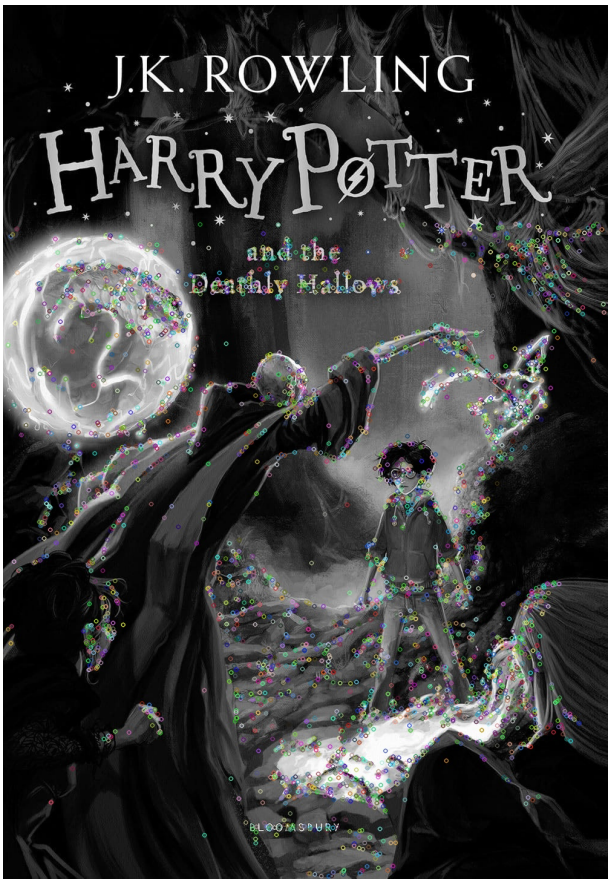

Figure 4. Extracted descriptors and keypoints of a train image using SIFT algorithm

2.3. Extracting features from train images

Using SIFT algorithm to find the keypoints and descriptors of each train image and storing it in separate lists. These keypoints and descriptors will be used later on for comparison with the camera frames. An example of identified descriptors is shown in Fig. 4 using a train image. Similar results are obtained for other train images as well (please view the attached results).

2.4. Comparison of train images features against camera-frame descriptors

Camera is turned on and each frame is used to detect the features (using SIFT) within the frame. These descriptors are compared with the train images feature dataset. Brute force matcher is used to get k (=2) best matches between the frame and train images. Knn technique is deployed to find matches (cross-validated) and ratio test is applied. An example of matches between camera-frame and train image is shown in Fig. 5.

Figure 5. Matches between camera-frame and a train image

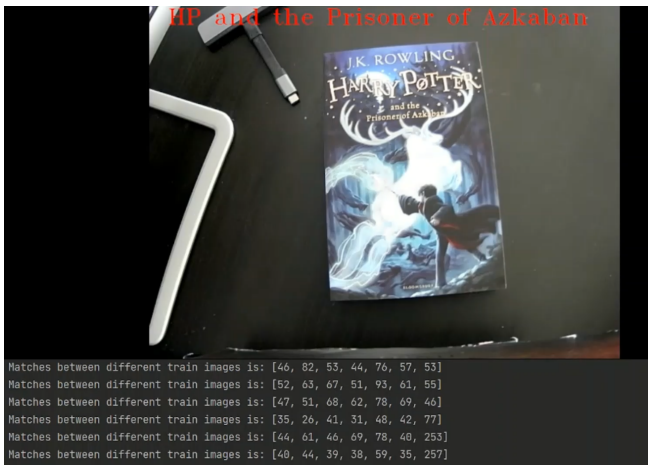

Figure 6. Result: Identified Book Cover

2.5. Thresholding to attain the best match

Thresholding technique is used to eliminate the train images with low matches. After finding the good matches between the train images and camera-feed, a counter is used to determine the number of good matches of each train image. The image with count value above the threshold and with the highest count is determined to be a good match for the book cover. An example of this result is shown in Fig. 6. (Please watch the video in Step 5 to find other results)

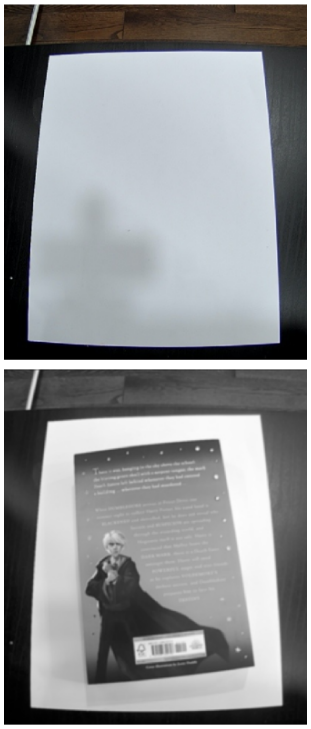

Figure 7. Gaussian Filtered Frame

Figure 8. Edges Detected in the frame

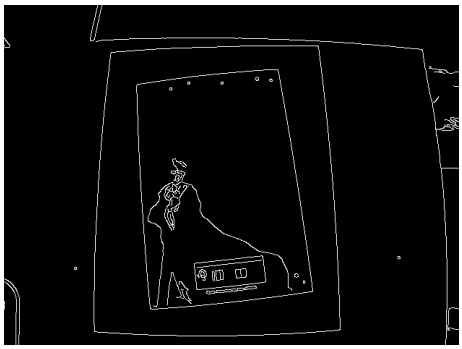

Figure 9. Morphological Operations on threshold image

Determining book dimensions and number of pages method

Following sections explain how the book dimensions and number of pages were determined.

2.3-b Smoothing Filter

After the steps of 2.2. Resizing and Grayscale, gaussian filter of kernel size equal to 5 is applied to remove noise. Result is shown for a frame in Fig. 7.

2.4-b Binarizing Frame

Canny edge detector is applied to binarize the frame image. To determine optimal Canny edge detectors threshold values, sigma value and median of the image are used and thereby, low and high threshold are calculated. The idea is to pick values that are close to the median. Result is shown for a frame in Fig. 8.

2.5-b Morphological Operations

Morphological operators are applied to remove unwanted blobs and ultimately, clean up the threshold image from previous step. Iterations of dilate and erode are performed and result is shown for a frame in Fig. 9.

2.6-b Finding external contours and bounding box

To be able to figure out the size of the book, we need an object in the frame that will allow us to determine the relationship between number of pixels and distance in 'millimeter' in the frame. To achieve this, I am using an A4 paper (size 210mm x 297mm), and book is placed on this paper to correctly measure the size. Area parameter is used to find the contour in the frame. For this setup to work, we need to ensure that the depth is correctly perceived by the camera and my approach to tackle this issue will be explained later. Contour and bounding box results are shown for a frame in Fig. 10.

2.7-b Warp image

From Fig. 10, we can clearly see that that camera is not aligned properly. Based on the bounding box, we can state that through the camera's eye the width and height is not consistent. To fix this issue, we need to warp the image by calculating transformation matrix and applying warp perspective transformation on the image. Result is shown for a frame in Fig. 11.

2.8-b Internal Contour

Now, we can find the internal contour i.e., the book on the A4 page. Result is shown for a frame in Fig. 12.

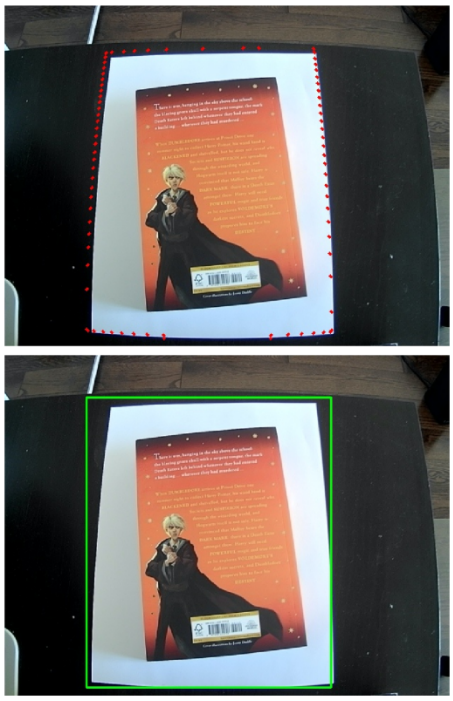

Figure 10. Contour and Bounding Box of A4 paper

Figure 11. Warped Perspective of A4 paper

2.9-b Determine all three dimensions of the book and compute number of pages

To find the height and width of the image, we can use the height and width parameters of the bounding box of the contour. Result is shown in Fig. 13.

However, to measure the thickness of the book, we need to account for the elevation of the book contour. Since the dimensions of the object changes as we move the object closer to the camera, we need to account for the elevation bias when the book is not laying flat as shown in Fig. 12. With the current setting, we can precisely measure the width

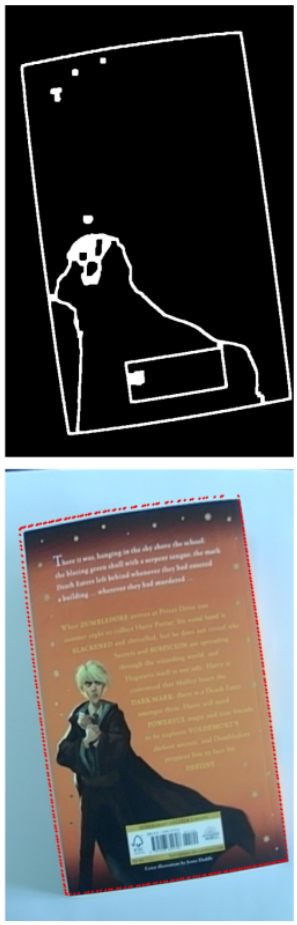

Figure 12. Book Contour on A4 paper

and height of the book as shown in Fig. 12, because the elevation in the contour is negligible. In this case, the thickness of the book does not play a significant role in skewing the depth perception of the camera. But when we place the book along its length and thickness against the A4 paper (as shown in Fig. 14), the width of the book produces an elevation bias. Measuring dimensions of an object requires geometric manipulation as explained below.

As explained earlier, in Fig. 14 the contour is lifted higher and the contour seen by the camera has a larger area than the actual area of the book on the A4 paper. To solve this problem, we need to analyze Fig. 15 and we can deduce the following: (let's say height/2 = h/2)

$$\frac{x}{d} = tan\beta$$

$$\frac{h/2}{d+w} = tan\alpha$$

For my case, d = 31" and (d + w) = 36". In reality, h/2 is equal to x, therefore,

$$31.tan\beta = 36.tan\alpha$$

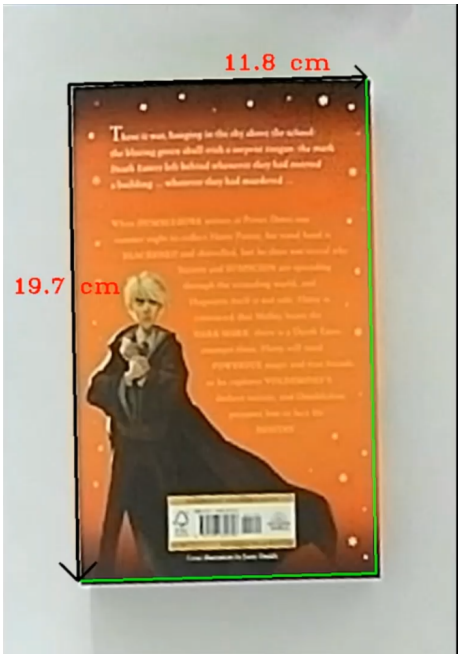

Figure 13. Result: Height and Width of the book

Figure 14. Height and Thickness side of the book

$$\frac{tan\alpha}{tan\beta} = 0.86 = 0.9(approx.)$$

Hence, I am using an elevation bias of 0.9 to correct the final output, because as 'd' grows, $\frac{d}{d+w}$ approaches value of 1. We can imagine 'x' is calculated as a large number and therefore, we need to reduce it by multiplying by elevation bias of 0.9. Final, result is shown along with the number of pages of the book in Fig. 16. To find number of pages, I determined the ratio of number of pixels along the height and used this ratio to find the number of pages per millimeter. To visualize the results to further details, please watch the videos attached with the report on Avenue to learn.

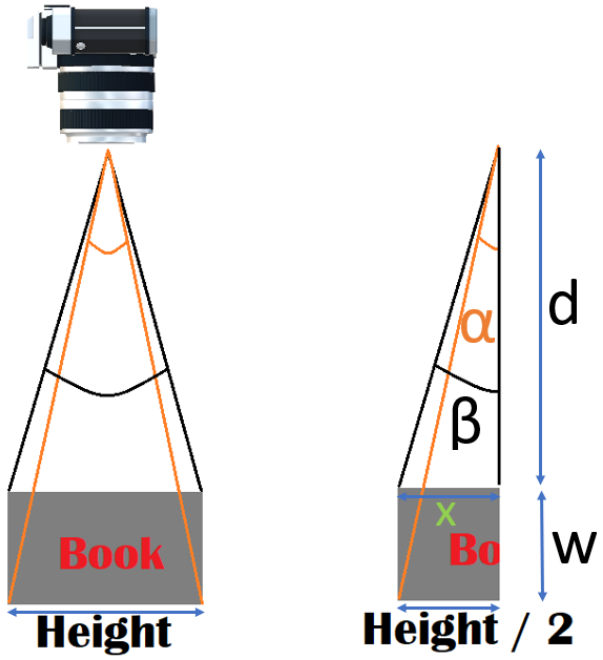

Figure 15. Camera Depth Perception

Figure 16. Result: Measured thickness, height, and number of pages

3. Experimental Results for Comparison with Proposed Method

To compare my method, I have implemented book cover recognition task using ANN and CNN. Results of these will

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten (Flatten)           (None, 2500)              0

 dense (Dense)               (None, 3000)              7503000

 dense_1 (Dense)             (None, 1000)              3001000

 dense_2 (Dense)             (None, 500)               500500

=================================================================
Total params: 11,004,500
Trainable params: 11,004,500
Non-trainable params: 0
```

Figure 17. ANN Model Summary

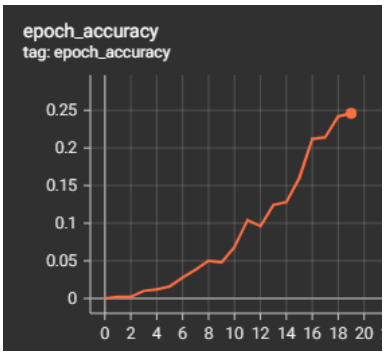

Figure 18. ANN Model Accuracy with 20 epochs

be presented in the following sections. Note that these experiments were conducted to understand the concepts behind the implementation, process of implementation and to compare the accuracy between different methods.
Book cover dataset has been collected from Kaggle [1] and only a portion of the dataset was used to compare the performance (500 train images) to quickly train the models.

3.1. Artificial Neural Network (ANN)

Fig. 17 shows the model that was created to test the accuracy and loss metrics of ANN. Only four layers are used with labels represented as sparse categorical.

Results of accuracy and loss can be visualized in Fig. 18 and Fig. 24 respectively. After 20 epochs, the model reached accuracy of 25.6% with loss of 5.4649 and while testing the model with test images, the model performed at accuracy rate of 45.0%. Similar images were used to test the trained model, however another challenge for the model could be developed by altering the scale and orientation of the test images. Result of test image is shown in Fig. 20 and Fig. 21, in this case model classified correctly and incorrectly respectively.

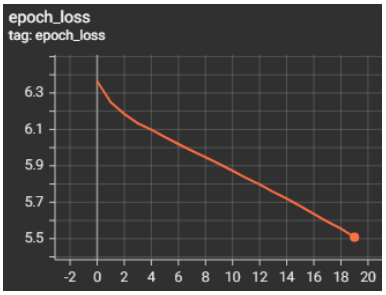

Figure 19. ANN Model Loss with 20 epochs

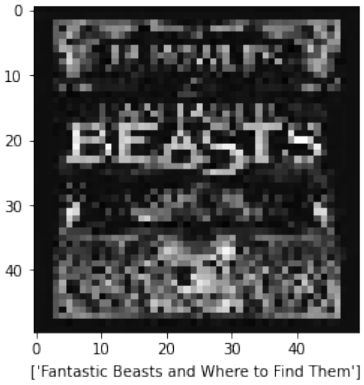

Figure 20. Result: Correct - ANN Test Classification

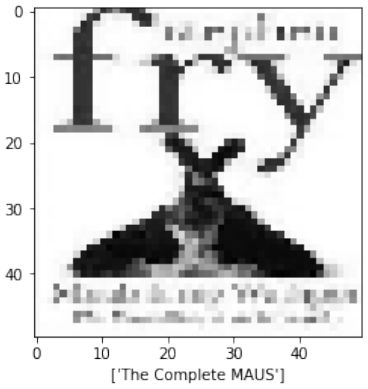

Figure 21. Result: Incorrect - ANN Test Classification

3.2. Convolutional Neural Network (CNN)

Fig. 22 shows the model that was created to test the accuracy and loss metrics of CNN. Only seven layers consisting of convolution, pooling, flatten and dense layers are used with labels represented as sparse categorical.

Results of accuracy and loss can be visualized in Fig. 23 and Fig. 24 respectively. After 20 epochs, the model reached accuracy of 88.8% with loss of 0.4668 and while testing the model with test images, the model performed at

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 48, 48, 32)        320

 max_pooling2d (MaxPooling2D  (None, 24, 24, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 22, 22, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 11, 11, 64)       0
 2D)

 flatten_1 (Flatten)         (None, 7744)              0

 dense_3 (Dense)             (None, 64)                495680

 dense_4 (Dense)             (None, 500)               32500

=================================================================
Total params: 546,996
Trainable params: 546,996
Non-trainable params: 0
```

Figure 22. CNN Model Summary

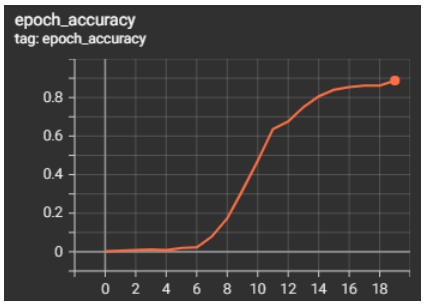

Figure 23. CNN Model Accuracy with 20 epochs

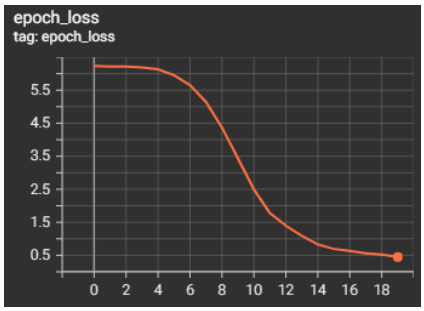

Figure 24. CNN Model Loss with 20 epochs

accuracy rate of 100.0%. Similar images were used to test the trained model, however another challenge for the model could be developed by altering the scale and orientation of the test images. Result of test image is shown in Fig. 25, in this case model classified correctly.

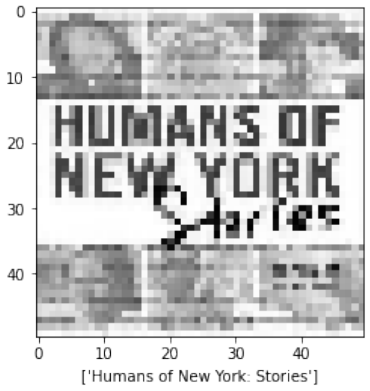

Figure 25. Result: Correct CNN Test Classification

4. Conclusion and Future Work

Utilization of SIFT has proven to be a strong candidate for feature detection in the cover pages of the books. However, similar to training CNN and ANN models, SIFT algorithm requires time to extract the features. An alternative is to extract and store the descriptors and keypoints for the entire dataset. In the future, a fruitful addition would be to incorporate alphanumeric character recognition techniques such as OCR as well. On the other side, as we have seen that CNN can clearly outperform ANN in almost all aspects. However, aside from the time complexity, a drawback of ANN and CNN is that it is not scale and orientation invariant. To improve the identification of book covers regardless of scale and orientation, we need to perform tasks of data augmentation and appropriately train the ANN and CNN models. In this project, my focus was to correctly recognize the book covers and determine the dimensions and number of pages of the book. These requirements were satisfied. In the future, another approach could be to not only link the book cover with its label but also, associate it with price, genre, author and other relevant information.

References

[1] LUKA ANICIN. Book covers dataset. https://www.kaggle.com/datasets/lukaanicin/book-covers-dataset. 7

[2] Tsai S. S. Schroth G. Chen D. M. Grzeszczuk R. Girod B. Chen, H. Robust text detection in natural images with edge-enhanced maximally stable extremal regions. (2011, September) In 2011 18th IEEE International Conference on Image Processing (pp. 2609-2612). IEEE.

[3] codebasics. deep-learning-keras-tf-tutorial. https://github.com/codebasics/deep-learning-keras-tf-tutorial/blob/master/16_cnn_cifar10_small_image_classification/cnn_cifar10_dataset.ipynb.

[4] Harrison@pythonprogramming.net. Classifying cats vs dogs with a convolutional neural network on kaggle. https://pythonprogramming.net/convolutional-neural-network-kats-vs-dogs-machine-learning-tutorial/.

[5] OpenCV-Python. Feature matching. https://opencv24-python-tutorials.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_matcher/py_matcher.html. 1

[6] R. Smith. An overview of the tesseract ocr engine. (2007). 2

[7] Linfeng Yang and Xinyu Shen. Book cover recognition. https://web.stanford.edu/class/ee368/Project_Autumn_1617/Reports/report_yang_shen.pdf. 1