Counting Actor Screen Time in Movies

Christina Lin
Department of Computer Science
Stanford University
chrlin@stanford.edu

Stephany Liu
Department of Computer Science
Stanford University
liu48185@stanford.edu

Samuel Wong
Department of Computer Science
Stanford University
samwwong@stanford.edu

Abstract

We present an end-to-end system for counting actor screen time in movies through detecting and clustering faces in Avengers trailers. We break the problem apart into two parts: face segmentation and facial recognition. Our data consists of trailers that we divide into different frames (images). Then, we solve the first part of the problem using state-of-the-art pre-trained models in order to detect bounding boxes around where the Avengers' faces are. Given this output, we feed the information into the second model which detects which Avenger character it is. We present results the show our quality predictions with different models.

1 Introduction

In the 220 minutes of Avengers: Endgame, the highest grossing blockbuster of all time, which characters and stories were given the most minutes to deliver a performance worthy of entertainment news cycles, word of mouth reviews, and pop culture canon? To satisfy this curiosity, we apply deep learning and CNNs to Avengers movie trailers to breakdown the screen time allocated to each actor or character. For our project, we will build an application to recognize faces and count how many seconds each Avengers character appears in a single video.

Our model consists of two modules: face detection, which draws a bounding box around each face in an image, and face recognition, which labels the face with an identity. For our baseline, we adapted OpenFace, which uses OpenCV for the first module and FaceNet for the second.

2 Related work

Although we were unable to find any prior research on our exact use case, we were able to find related works for similar movie facial segmentation and object detection problems. For actors' facial segmentation, the three papers we found are as follows: Movie Scene Segmentation Using Object Detection and Set Theory, [3] End-To-End Face Detection and Cast Grouping in Movies Using Erdos-Renyi Clustering, [4] and Towards an Automatic Face Indexing System for Actor-based Video Services in an IPTV Environment. [2] What we liked about the first paper was how they segmented the entire movie into scenes before performing deep learning algorithms. In addition, all three papers optimized for few false positives, with their metric being weighted more heavily toward precision than recall. What we decided to do was to have a more balanced approach to our metric, as too much weight toward precision can cause many true positive images to not be tagged. The most clever approach was in the Erdos Renyi clustering paper, where the large clusters of point pairs in the face can be fully connected by joining just a small fraction of their point pairs, while small clusters, which would indicate different people, would lead to worse results. This technique is state-of-the-art and sets this paper apart from the others.

For object detection, we will look into popular algorithms such as Faster R-CNN and YOLO. [5] YOLO has the benefit of reducing computation cost by only looking once through the image while maintaining accurate predictions. The details of the YOLO architecture are explained in section 4, as we did use the YOLO model in our project.

To tackle the challenges such as the recognition of hard faces, we can follow past examples such as SFA and Robust Face Detection via Learning Small Faces on Hard Images. [7]

3 Dataset and Features

3.1 Training Data

To train a facial recognition model that can identify Avengers characters, we prepared a dataset with around 100 images for each character that appears in the franchise. The images were scraped from a Google Image search for a concatenation of the actor name and character name.

Representing all of the major characters that appear in the first two Avengers movies, our training data includes the following characters: Loki (Tom Hiddleston), Nick Fury (Samuel L Jackson), Hawkeye (Jeremy Renner), Black Widow (Scarlett Johansson), Iron Man (Robert Downey Jr), Captain America (Chris Evans), Thor (Chris Hemsworth), Hulk (Mark Ruffalo), Scarlet Witch (Elizabeth Olsen), Quicksilver (Aaron Taylor Johnson). Our dataset contains around 1000 images. Since we are fine-tuning a model that has already been trained on over 6000 pictures, this relatively small size is still fairly effective.

3.2 Dev/Test Data

For testing, we have prepared four Avengers movie trailers, each transformed from video into images captured from the frames of the video at one second intervals. These trailers are: The Avengers (2012), Avengers: Age of Ultron, Avengers: Infinity War, and Avengers: Endgame. Each trailer is around two minutes, which results in roughly 120 images per video. Each image is labeled with the number of faces that appear, and each face is labeled with the name of the character.

4 Methods

4.1 Preprocessing

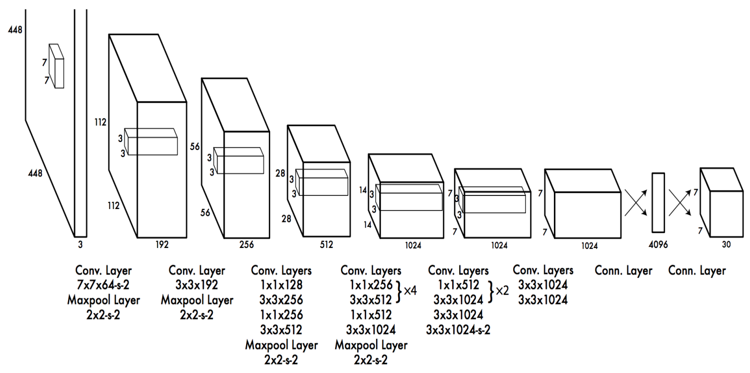
As input our model takes images, not video, the first step is to convert movies into frames. As described above, we break down the video into frames taken one second apart. We then apply our deep learning model to each image in order to identify the faces that appear in each frame. The total screen time for each character is the sum of 1-second frames that they appear in.

4.2 Two-step approach

Our model consists of two steps. The first step is face detection or segmentation: examine each image and to find the bounding box around each different face in the image. The second step is facial recognition: identify who that character is.

4.3 Face Detection with OpenCV

OpenCV Face Recognition is a pretrained model provided by OpenCV for face detection trained on triplet loss. The entire OpenCV Face Recognition pipeline includes identifying the location of the face within an image, cropping it accordingly, and creating a 128-d embedding to quantify the face.



Conv. Layer 7x7x64-s-2 Maxpool Layer 2x2-s-2
Conv. Layer 3x3x192 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x128 3x3x256 1x1x256 3x3x512 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x256 3x3x512 ×4 1x1x512 3x3x1024 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x512 3x3x1024 ×2 3x3x1024 3x3x1024-s-2
Conv. Layers 3x3x1024 3x3x1024
Conn. Layer
Conn. Layer

Figure 1: https://papers.readthedocs.io/en/latest/imagedetection/yolo/

The implementation for this project uses a Haar-based cascade frontal face classifier. We applied the detector to our data and extracted the bounding box information obtained prior to the cropping, opting to use a separate model for facial similarity after the location of the face was detected.

4.4 Face Detection with YOLO

YOLO (You Only Look Once) is an object detection CNN consisting of convolution, maxpool, and fully connected layers. The architecture of this network is shown below. We used a pretrained YOLO model takes an RGB image as an input and ouptuts the location of the bounding box where the face (object) was detected. There are 24 convolutional layers (some with maxpool) followed by 2 fully connected layers. Finally, the model uses IOU (intersection over union) in order to improve the confidence of the predictions and well as remove possible duplicate boxes.

4.5 Face Detection with MTCNN

MTCNN (multitask cascaded convolutional networks) is a state-of-the-art facial detection model developed by Zhang et al. [9] This technique exploits the correlation between face detection and alignment in order to boost performance.

Our implementation uses the pretrained Keras FaceNet model by Hiroki Taniai, which uses an Inception-Resnet v1 model with softmax loss over the MS-Celeb-1M dataset [8]. To combat false positives, detected faces were post-processed using a confidence threshold hyperparameter before being fed into the classifier.

4.6 Face Recognition with FaceNet

FaceNet is a facial recognition model developed at Google by Schroff et al. [6] The original implementation trains embeddings using a deep neural network with triplet loss.

Our baseline model uses OpenFace [1], an open-source Python and Torch implementation of FaceNet. OpenFace takes a two step approach to face recognition. The first step is to crop an image using bounding boxes to locate where the faces are in an image. OpenFace uses OpenCV, as described in 4.3 above.

The second step is to recognize who that face is. This is the same as our approach described above. The current OpenFace model is built to classify eleven specific celebrities, a list that does not include any of the actors in the Avengers series. Thus, we fine-tuned the facial recognition model with a training dataset of Avengers actors and actresses as described in Section 2.1.

4.7 Data Augmentation

Initially, as we were working with the training and dev set, through our error analysis we observed some misclassifications due to rotations and color imbalance. In order to increase the amount of training data for our model to help solve these issues, we used data augmentation techniques to

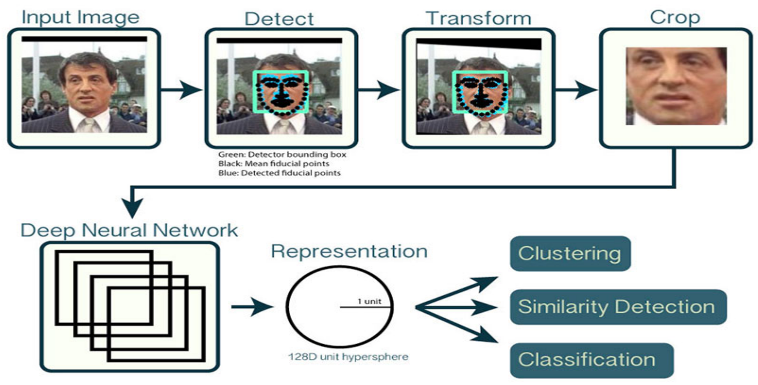


Figure 2: https://www.pyimagesearch.com/2018/09/24/opencv-face-recognition/

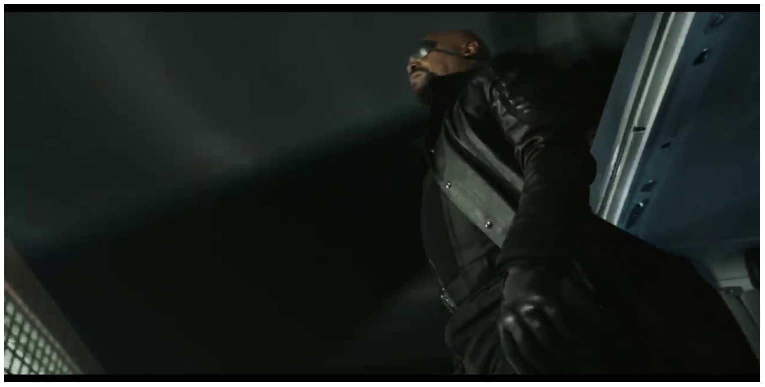


Figure 3: OpenCV face detection is unable to detect Nick Fury's face in Avengers 2012

enlarge our dataset. We decided on using 4 translations (up, left right, down), 4 rotations (different angles), 4 distortions, and 3 RGB color changes.

5 Experiments/Results/Discussion

Our primary metric compares the total screen time, hand-labeled by us, to the total screen time predicted by the model per actor. Secondarily, for each frame, we will examine the number of bounding boxes, or faces detected, to evaluate the performance of the first part of the model. For each bounding box, we will calculate the precision of the facial recognition prediction. Taking a weighted average of these scores per frame will tell us the performance of the initial part of the model vs. the latter part of the model.

We compared the performance of each of our models through a thorough frame-by-frame error analysis revealed. We found that OpenCV face detection often failed in two cases: dark, blurred faces, and angled or side profile faces. In contrast YOLO and MTCNN were much more sensitive in these cases and were able to draw correct bounding boxes on more faces. As shown in Table 1, YOLO and MTCNN achieved significantly higher accuracy than OpenCV. However, these two models were also more prone to false positives, detecting "faces" were there were not actually faces. An example of these different models is shown in Figure 3, where OpenCV fails to detect the face, and Figure 4, where YOLO is able to detect the face.

6 Conclusion/Future Work

Although face detection and face recognition seem to be well-studied, nearly solved problems in deep learning, they proved to be tricky to apply to real world problems such as this one. Face detection was difficult on images that were captured from movies, especially when frames were dark, blurry,

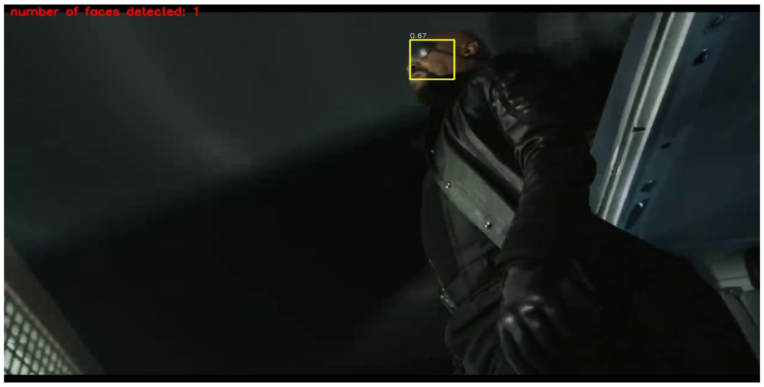


Figure 4: YOLO is able to detect Nick Fury's face in Avengers 2012

Table 1: Face detection accuracy from different models

	OpenCV	YOLO	MTCNN
Percentage of faces correctly detected (Step 1)	72%	93%	96%
Percentage of false positives	2.6%	7.0%	28%

and shot from dramatic angles. Face recognition was difficult when faces were contorted by dramatic expressions, and again, when the face is dark, blurred, or shown from a strange angle. When two models are combined to solve one problem, small errors compound to larger inaccuracies.

The individual models that we tried each seemed promising in different ways. In the future, we can try freezing and re-training different layers within the models to find a better balance between, for instance, extremely sensitive face detection, and too many false positives. In addition to exploring awe would like to try more hyperparameter tuning on a single model.

7 Code

Code is provided on the Github page located at this URL: https://github.com/schliu/cs230-avengers/

8 Contributions

Christina scraped and labeled training and test data, performed error analysis on outputs from each model, and ran the YOLO model. Stephany trained and ran the OpenFace, MTDNN + FaceNet, and YOLO + FaceNet models. Sam scraped and labeled test data, performed error analysis, and performed data augmentation. All team members contributed equally to this paper.

Table 2: Predicted screen time (in seconds) for Avengers 2012

	Human	OpenFace	MTCNN + FaceNet	YOLO + FaceNet
Screen time - Loki	2	4	5	7
Screen time - Fury	3	0	1	1
Screen time - Hawkeye	2	3	4	3
Screen time - Black Widow	5	3	5	4
Screen time - Iron Man	15	11	17	16
Screen time - Thor	2	3	8	9
Screen time - Captain America	5	6	13	9
Screen time - Hulk	1 2	4	3	
Screen time - Scarlet Witch	0	0	0	5
Screen time - Quicksilver	0	0	0	1

Table 3: Predicted screen time (in seconds) for Age of Ultron

	Human	OpenFace	MTCNN + FaceNet	YOLO + FaceNet
Screen time - Loki	0	0	1	1
Screen time - Fury	1	1	4	3
Screen time - Hawkeye	3	3	7	4
Screen time - Black Widow	6	7	4	6
Screen time - Iron Man	5	4	8	5
Screen time - Thor	3	3	3	3
Screen time - Captain America	8	3	10	9
Screen time - Hulk	7	6	6	4
Screen time - Scarlet Witch	4	0	0	3
Screen time - Quicksilver	1	0	0	5

References

[1] Brandon Amos, Bartosz Ludwiczuk, and Mahadev Satyanarayanan. Openface: A general-purpose face recognition library with mobile applications. Technical report, CMU-CS-16-118, CMU School of Computer Science, 2016.

[2] Jae Young Choi, Wesley De Neve, and Yong Man Ro. Towards an automatic face indexing system for actor-based video services in an iptv environment.

[3] Ijaz Ul Haq, Khan Muhammad, Tanveer Hussain, Soonil Kwon, Maleerat Sodanil, Sung Wook Baik, and Mi Young Lee. Movie scene segmentation using object detection and set theory. *International Journal of Distributed Sensor Networks*, 15(6):1550147719845277, 2019.

[4] SouYoung Jin, Hang Su, Chris Stauffer, and Erik Learned-Miller. End-to-end face detection and cast grouping in movies using erdős-rényi clustering. *ICCV 2017*.

[5] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[6] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

[7] Chris Stauffer Erik Learned-Miller SouYoung Jin, Hang Su. Robust face detection via learning small faces on hard images.

[8] Hiroki Taniai. keras-facenet. `https://github.com/nyoki-mtl/keras-facenet`, 2018.

[9] Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, and Yu Qiao. Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters*, 23(10):1499–1503, 2016.