Cartoon Facial Recognition

Ishita Goluguri (ishi) and Yajvan Ravan (yravan)

Abstract

In this project, we tackle the problem of cartoon face recognition. While face detection techniques for humans have been extensively studied, face recognition for cartoons has rarely been researched. One of the reasons for this is the large variance in cartoon styles and quality. Because of this, we augment our data with a variety of features, such as blur, brightness, and reflection, and analyze the effect of each of these data augmentations on different styles of cartoons. The core algorithm in our methodology was the EigenFaces face recognition algorithm, which uses a statistical technique to represent images as a linear combination of eigenvectors and compute a distance between training and test images to assign a predicted label to the test image. The blurring augmentation performs the best to improve EigenFaces accuracy drastically

1. Introduction

Cartoon face recognition is an essential aspect of understanding and interacting with the virtual world. It plays an important role in various commercial applications, including automatic cartoon face verification, computer-aided cartoon face generation, and cartoon movie recommendation. While models have achieved significant success in recognizing and verifying human faces using large-scale face recognition datasets, cartoon character recognition lags behind.

Recognizing cartoon faces poses unique challenges due to their intrinsic characteristics. Firstly, cartoon faces often have smooth and sparse color regions, limiting the availability of texture information. In addition, they have fewer shadows and lines, which makes features more difficult to detect. Secondly, the details of cartoon faces are predominantly defined by their structure and shape, which are characterized by sharp and clear edges. Therefore, accurately identifying the sparse and critical shape patterns is crucial for distinguishing different cartoon characters.

To tackle this problem, we use a face recognition algorithm, EigenFaces, that is more commonly used on people. To our knowledge, there have been no cases of using this algorithm in comics. EigenFaces utilizes Principal Component Analysis (PCA) to extract the significant facial characteristics. It represents faces as a linear combination of EigenFaces, which are determined by the eigenvectors of the covariance matrix derived from the face dataset. EigenFaces effectively captures the inherent variations in face images, enabling efficient recognition and reconstruction. By projecting a new face onto the eigenface basis, it can be compared to existing EigenFaces to determine the closest match.

This representation is also concise and doesn't take as much computation power as a CNN, for instance. Where a CNN might take hours to train on a given dataset, the Eigenfaces algorithm can be trained and run in mere minutes. This makes it far better in terms of computation.

To tackle the additional challenges that come with using cartoons, we use a variety of data augmentation techniques. The dataset used, iCartoonFace [5], has images with low lighting, blurred images, and images with busy backgrounds. To combat this, our implementation of EigenFaces uses data augmentation, and we study a variety of different techniques on different categories of character types. In summary, the contribution of this work is to implement EigenFaces on cartoon characters for the first time, as far as we know, and to use data augmentation and other data processing methods to increase the accuracy of the algorithm.

2. Related Work

Much of the work in the area of cartoon/comic face detection has been limited. One of the reasons, is that there are few straightforward commercial applications for models that can detect and recognize cartoon characters with high accuracy. However, such applications do exist and are becoming more popular as the animation industry grows. Such applications are in image search engine optimization or advertising with cartoon characters. Secondly, large, standardized data sets of cartoon characters and faces did not exist until a few years ago, and are still not very abundant.

2.1. Datasets

Two primary datasets for face/character recognition exist. Toonnet [10] contains a few thousand cartoon-like images. The more recently released iCartoonFace dataset

[9] contains around 400,000 images with bounding boxes, pose, and character identity annotations. We used the latter dataset for our work.

2.2. Character Recognition/Detection

Early works, before the aforementioned datasets, relied on feature detection methods, particularly by Takayama et. al. [6]. They tried a wide variety of methods to segment faces from the background, such as skin color and skin edge extraction, jaw contours, and hair extraction. These methods had around 50% precision, and would not generalize to non-humanoid character that have non-standard facial features.

More recent works have relied on various deep learning models. For example, the Manga FaceNet achieved 97% precision on Japanese comics and cartoons [2]. Recently, a team demonstrated facial detection and recognition using the MTCNN architecture with 80% precision [4]. However, one shortcoming of that study was the use of a limited 100 image database, as iCartoonFace was not released at the time. The most recent paper dealing with cartoon face recognition used a Graph Convolutional Network along with a graph jigsaw puzzle to achieve 80-90% accuracy on a large subset of iCartoonFace [5]. Aside from these few examples, very little research has been done in this area.

The approach we propose is centered around the EigenFaces algorithm proposed in [8]. The main advanatages of using such an algorithm lie in its simplicity compared to neural network models. The algorithm works by constructing an eigenfunction decomposition of the training distribution. In essence, we use the image database and extract eigenvectors from its covariance matrix to do Principle Component Analysis(PCA). These vectors form a basis of "EigenFaces", and we can compare decompositions of test images to training images for quick inference and classification. This algorithm requires significantly less training time, no hyperparameter tuning, and very little memory usage compared to large NN models. This algorithm was proposed specifically for face recognition, achieving up to 96% accuracy in [8]. Most modern face detection methods use some variation of EigenFaces or PCA.

As such, given the increased variability, and therefore separability, of cartoon characters, we expect that EigenFaces should perform similarly on cartoons.

3. Methodology

3.1. Hypothesis

We wish to show the viability of the EigenFaces algorithm on the iCartoonFace dataset for face recognition. Furthermore, we wish to improve its accuracy with multiple forms of data augmentation, partiuculary through instance segmentation. Finally, we wish to compare the improvements of various data augmentations on various types of cartoon characters, for example by looking at the difference that blurring makes on sharply drawn characters vs softer, textured characters.

3.2. Dataset

The EigenFaces Algorithm was implemented on the iCartoonFace dataset [9], which contains images of thousands of characters, from anime to animals to Spiderman. The characters are pulled from a variety of cartoons, and the images of the character are pulled from a variety of sources, incuding TV cartoons, comic books, ads or posters, and fanart. This leads to many different styles of images in characteristics such as brightness, blurriness, and background. We depict a few images from this dataset below **??** to highlight the amount of variance in a single character.

In order to limit training time and data, we picked 15 distinct characters from a randomly selected folder in iCartoonFace. Each character had 60 images of them in various poses, lighting, and backgrounds, and of these we selected the remaining images to be our test set.

3.3. EigenFaces

Our methodolgy centers around usage of the EigenFaces face recognition algorithm with implementation inspired by [7].

To begin, we preprocess the training dataset by loading the images in grayscale and putting them into an array; we put all labels into another array. Each image is 100 pixels by 100 pixels. After, we flatten each image into a one-dimensional vector. Next, we apply Principal Component Analysis, a method which is used to minimize the number of parameters we use to describe data.

We compute the sample mean, and then compute the covariance matrix of the data and its corresponding eigenvectors/EigenFaces. We then sort the eigenvectors, and pick the k most desdriptive ones, where we have used $k = 16$ for most of our results. We can then encode every training image into a vector of length k, i.e linear decomposition, and these are known as the k principal components.

Finally, we perform recognition on test images by finding the nearest neighbor between the decomposed test image and the decomposed training images. This represents our general training + inference pipeline for the EigenFaces algorithm.

3.4. Data Augmentation

We use 4 primary data augmentations: blurring, brightness, segmentation, and flipping.

Blurring is implemented using a 10 by 10 box filter. This has the effect of removing sharp features across a large neighborhood. We predicted that this would have the effect

(a) Originapl Image (b) Segmented Image

Figure 1. Image Segmentation

of improving accuracy among all characters. This is because, given the fact that this method uses linear decomposition, small changes in the location of sharp features would lead to drastically different decompositions, and blurring removes this effect of spatial variance.

Brightness is both decreased and increased respectively, with contrast control, using the following transformations on the image (which has grayscale pixel values from 0-255):

$$\left[image\right] \longrightarrow 0.75\left[image\right] - 50$$

$$\left[image\right] \longrightarrow 1.25\left[image\right] + 50$$

We expect brightness augmentation to minimally affect the accuracy, since the EigenFaces algorithm performs linear decomposition, and brightness augmentation has the effect of linearly shifting the dataset, with some small nonlinearity introduced by clipping the pixel values between 0 and 255.

Flipping is done both horizontally and vertically. We expect this to improve accuracy for much of the same reasons as blurring, and because there will simply be more data to train on.

Segmentation is explained below.

3.5. Segmentation

We wish to use instance segmentation models as another data augmentation to improve. The idea is that perfectly performed instance segmentation will separate the character from the background, limiting literal background noise, and maximizing the difference between characters.

We perform segmentation by using a pre-trained Mask-RCNN [3] [1]. An example of this segmentation is shown in Figure 1. We run the model on all of our training images to create alternate data images with just the characters, and then we use the segmented images to train a new version of EigenFaces. We then perform the same data augmentations as above on the modified dataset and compare final performance on the test set.

3.6. Evaluation

We try out multiple combinations of training EigenFaces and compare the performance of all of these methods. EigenFaces alone will form our baseline. We then perform the same training with EigenFaces and the various augmentations of our training data as explained above. Finally, we test each of the trained algorithms on the same test set. For each, we compute and compare the test accuracy for each character class. We also compare the accuracy of the best augmentation over multiple characters too see how different characters are affected by this augmentation.

Furthermore, we perform an ablation study, varying the number of eigenvectors (k) used to decompose the dataset. The larger the number of eigenvectors, the greater test accuracy one will get, but the more data will be required to store it.

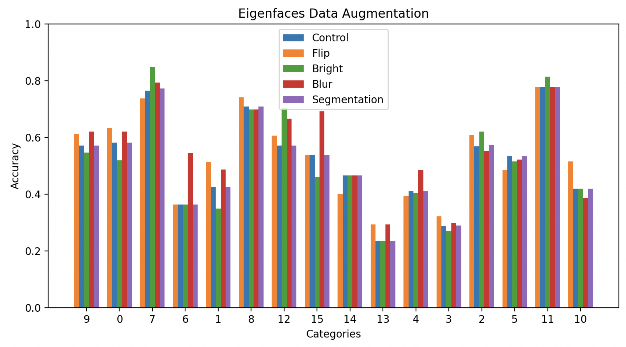

Figure 2. Randomly Chosen Characters After Variety of Data Augmentation

4. Results

The primary focus of our project is data augmentation as a way of increasing accuracy with EigenFaces on cartoon characters, so in the evaluation, we focused on the four primary types of data augmentation: blur, brightness, segmentation, and flip. In addition, we chose fifteen arbitrary characters out of the iCartoonFace dataset for efficient testing. These characters cover a large range of styles and characters. They include anime, cartoons, and drawing style. In addition, one of the characters is a dog, and several are nonhuman machines. In addition, there are images of varying angles, lighting, crop, and quality.

For each character, the data was split into 60 images used for training and the remainder used for testing. The results of the data augmentation testing can be seen in figure 2. Each of these data augmentations took the EigenFaces algorithm about two minutes to train and then test on the entire remainder of the dataset. As expected, there were differing results for each type of data augmentation depending on the style of image.

The first thing to note in the results of our experiment is that blurring was the best augmentation, improving accuracy on 11/15 characters. This is likewise due to the removal of the spatial noise from sharp features moving around, as hypothesized above. Some characters had large improvements in accuracy up to 15%, notably characters 6 and 15.

The next best augmentation was flipping. This improved the test accuracy of 8 out of 15 characters. However, any improvements were no more than 7% and for a few characters. One likely reason for this is that, after flipping the images, the dataset becomes significantly more complex. There is no reason that the flipped images are close to the originals in hyperspace, unlike after blurring. Thus, it becomes harder to linearly separate the different characters, and therefore, the new dataset likely takes more eigenvectors to perform better. Hence, this also explains why performance may in fact decrease after this augmentation

The third best augmentation was brightness manipulation. We note that this only improved the accuracy of 4 out of 15 characters. The reason for this poor performance likely lies in the explanation outlined above. Because brightness augmentation is a linear shift, this augmentation distinctly moves the images in hyperspace and therefore makes it much harder to separate the characters. Unlike flipping, where the movement in hyperspace is largely variable, brightness augmentation shifts the images directly, and therefore has worse performance.

Finally, segmentation had no effect on any of the characters. This result is likely rationalized by noting that the backgrounds of the characters are only a minor portion of the image, and much of the image is the character face. Therefore, we hypothesize that although segmentation makes the dataset more linearly separable, the effect is not pronounced enough to significantly affect test accuracy.

These results led to a natural question of which types of augmentations worked best for which style of cartoon character. Since a single cartoon is consistent with style throughout, gaining insight into this question is very impactful since it would allow EigenFaces, a much less computationally intense method than a neural network, to be used to extract faces from a cartoon. Due to time constraints, we chose to focus on the blur data augmentation. Looking at Characters 6 and 15, which had the most drastic accuracy increase from the blur data augmentation, we hypothesized that characters with fewer details but crisp lines would respond the best to this data augmentation. We chose four such characters.

The resulting accuracy graph, shown in figure 3, didn't show as drastic of an increase as the two original testing categories, but most of them showed some increase with the blur augmentation. This lines up with our hypothesis.

Finally, we ran an ablation study to characterize the effect of the single hyperparameter, k, on the performance of this algorithm. We note that as k increases, the space required to describe the dataset increases, as does the time of inference. However, a larger k also captures more information in the data, and prevents loss of information from projecting onto a small subspace.

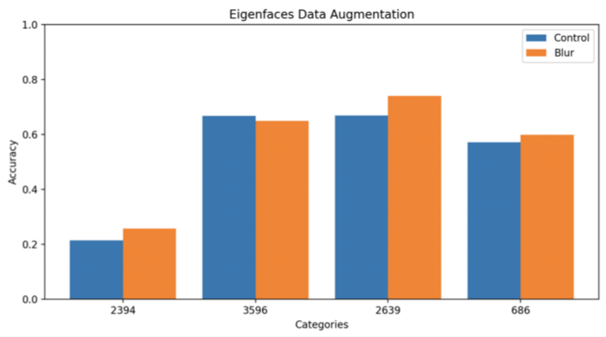

Figure 3. Specifically Chosen Characters After Blur Augmentation

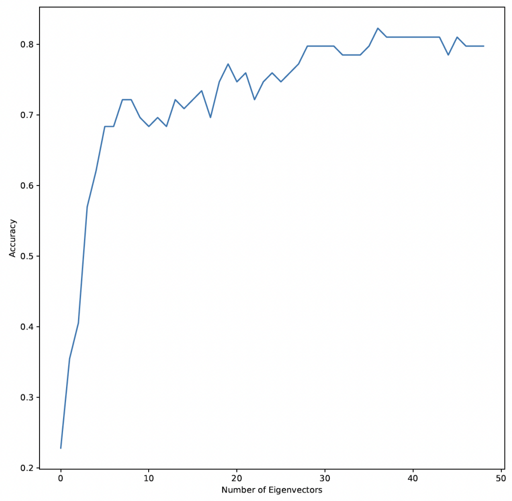

Figure 4. Ablation Study

The results of the ablation study show that the optimal k is around 40. Increasing k past this leads to marginal improvements in the test accuracy.

One strong limitation of these experiments is the dataset. The EigenFaces algorithm is highly dependent on the training dataset. Thus, the limited number of character classes in our dataset, may have an effect on how the augmentations affect the accuracy. Another limitation lies in the linear separability of the dataset. If a few charactere classes are not very linearly separable in hyperspace, this can decrease the accuracy of the algorithm drastically, especially as the number of EigenFaces decreases.

Much work has not been done in the field of cartoon face recognition, and the work that has been done has used neural networks. Other studies using neural networks have approached accuracy of up to 0.75. On average, our results indicate that the EigenFaces algorithm achieves around 50% accuracy without data agumentation. The blurring augmentation improves this by around 5%. However, this is still far from the accuracy achieved by CNNs. Yet, EigenFaces carries so many advantages in terms of training time and computational resources that improving it to reach this benchmark is a worthwhile prospect, and our results indicate that thoughtfully chosen data augmentations for specific characters can improve test accuracy significantly.

5. Conclusion

Thus, as we can see from the results, data augmentation had mixed effects on the performance of the EigenFaces algorithm. The original accuracy of this algorithm is around 50% and the only augmentation that significantly improved accuracy was blurring.

The explanations for this lie in the inner workings of the algorithm. EigenFaces works by projecting images onto a subspace defined by the eigenvectors of the training dataset, i.e. linearly decomposing test images. The more linearly separable the characters in the training set, the fewer eigenvectors are required, and the better accuracy is achieved. Blurring removes small variances in spatial locality of sharp features, and therefore moves these images closer together in hyperspace, improving accuracy. This was demonstrated by testing on specific characters with such features and few other features. Flipping moves images drastically in hyperspace in a way that is highly variable, and therefore does not have much positive effect. Likewise, brightness augmentation, as a linear transformation separates images that we want to be classified together, and segmentation does not have enough of an effect on the separability.

Thus, data augmentation shows some promise in how it can affect the performance of EigenFaces. Thus, theres is significant prospect in using this algorithm and improving it to be on par with existing neural network methods for the problem of cartoon face recognition.

A future direction we'd like to take with our work is to explore the blur data augmentation more, with various kernels. We also want to explore which other augmentations work best for other types of characters. In addition, analyzing other augmentations on the sampling of images chosen to try the blur augmentation may reveal interesting results. Finally, we wish to test the effect that the size of the training dataset has on this algorithm.

There are many open questions in this problem. The efficacy of neural network methods for cartoon face recognition is poor compared to other domains, and the effectvieness of other methods, or combinations of methods is yet to be explored. We have attempted to explore and improve another method in this work, and our results indicate that there is promise in using EigenFaces.

References

[1] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019. 3

[2] Wei-Ta Chu and Wei-Wei Li. Manga facenet: Face detection in manga based on deep neural network. In *Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval*, ICMR '17, page 412–415, New York, NY, USA, 2017. Association for Computing Machinery. 2

[3] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn, 2018. 3

[4] Saurav Jha, Nikhil Agarwal, and Suneeta Agarwal. Towards improved cartoon face detection and recognition systems. *CoRR*, abs/1804.01753, 2018. 2

[5] Yong Li, Lingjie Lao, Zhen Cui, Shiguang Shan, and Jian Yang. Graph jigsaw learning for cartoon face recognition. *CoRR*, abs/2107.06532, 2021. 1, 2

[6] Kohei Takayama, Henry Johan, and Tomoyuki Nishita. Face detection and face recognition of cartoon characters using feature extraction. 2012. 2

[7] Svetlana Topalova. eigenfaces. https://github.com/svetlana-topalova/eigenfaces/tree/master, 2019. 2

[8] M.A. Turk and A.P. Pentland. Face recognition using eigenfaces. In *Proceedings. 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 586–591, 1991. 2

[9] Yi Zheng, Yifan Zhao, Mengyuan Ren, He Yan, Xiangju Lu, Junhui Liu, and Jia Li. Cartoon face recognition: A benchmark dataset. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 2264–2272, 2020. 2

[10] Yanqing Zhou, Yongxu Jin, Anqi Luo, Szeyu Chan, Xiangyun Xiao, and Xubo Yang. Toonnet: A cartoon image dataset and a dnn-based semantic classification system. In *Proceedings of the 16th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry*, VRCAI '18, New York, NY, USA, 2018. Association for Computing Machinery. 1