Anime face recognition applied by ensemble learning and transfer learning

陈奕铭

2020103006

International School, Jinan University

1 Introduction

In this paper, we approach the problem of classification using random forest and the inception V3. We use Inception V3 as the main algorithm in this paper and compare this result with the result from the random forest. Our goal is to apply these two algorithms to classify some of the anime characters to output the character name of their face image input. The image inputs are usually from anime snapshots, some emoji, or some illustrations from other painters that have been uploaded to some illustration website like Pixiv. We build our dataset, having 21 characters with a total of 3909 images of faces. And we used 80% dataset as the training in both random forest and the inception V3. Finally, we achieve 89.1% accuracy with Inception V3 and image augmentation.

2 Previous work

2.1 Random forest

Random forest is the implementation of stochastic discrimination classification methods. For classification tasks, the output of the random forest is the class selected by most decision trees. The decision tree can produce inspectable models and is also robust, but the accuracy is low. Because decision trees are easy to become overfit to the training sets, having very high variance. Random forest is used to improve the final result by training on different parts of the dataset in different decision trees. It can be averaging multiple deep decision trees, which improves the performance a lot with some increase in bias and some loss of interpretability. [1] In order to separate different individual trees to be not too correlated, we need to use bagging, which selects a random sample with a replacement of the training set. [2]

$$\begin{aligned}&\textit{Bagging steps}:\\&\textit{Giving a trainging set } X = x_1, \ldots, x_n \textit{ andresponses } Y = y_1, \ldots, y_n\\&\textit{For } B \textit{ times bagging } b = 1, \ldots, B:\\&1.\textit{Sample, with replacement, } n \textit{ training examples from } X,\ Y;\ \textit{call these } X_b, Y_b.\\&2.\textit{Train a classification or regression tree } f_b \textit{ on } X_b,\ Y_b.\end{aligned} \tag{1}$$

The predictions are made of the majority classification produced by all trees. Also, random forest use "feature bagging" in this process to reduce the correlation furtherly. For classification problems with p features, $\sqrt{p}$ is used in each split.

2.2 Inception V1

In ImageNet Large-Scale Visual Recognition Challenge 2014, a team at Google propose a deep convolutional neural network architecture Inception V1, which is 22 layers deep network. [3] The Inception V1 come out to avoid multiple deep layers of convolutions used in a model that resulted in the overfitting of the data. It uses multiple filters of different sizes on the same level, which means replacing deep layers with parallel layers. And this model is

made up of four parallel layers.

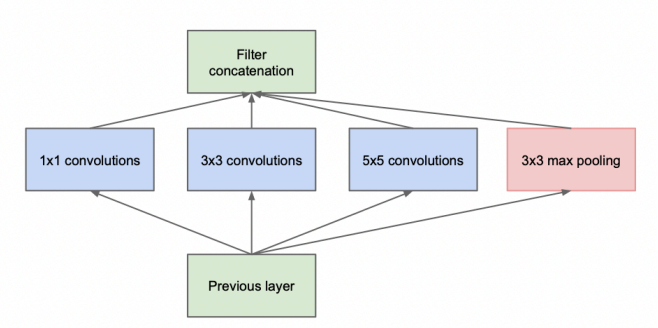


Figure 1. The original block

And to reduce the computation expenses, 1*1 convolutional layers are added before each convolutional layer, which speeds up the computations. And it becomes the module of Inception V1.

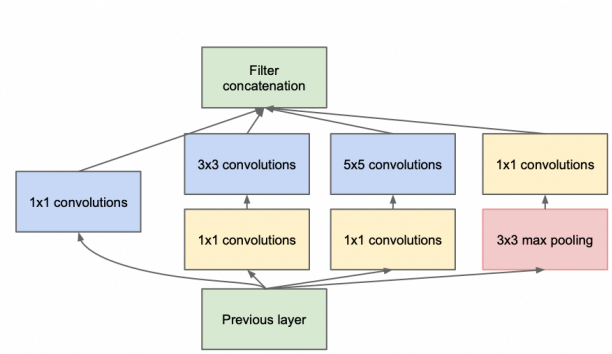


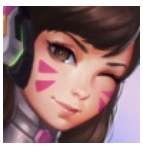
Figure 2. The real block used in Inception V1

3 Dataset construction

3.1 Overview of the dataset

The final dataset contains 21 characters and 3903 images, they are from 7 animes, 2 games, and youtube. The source, character name, and face images are listed in the table below.

Source	Character	Image
K-ON	Yui Hirasawa	
K-ON	Tsumugi Kotobuki	
K-ON	Ritsu Tainaka	

K-ON	Azusa Nakano	
K-ON	Mio Akiyama	
Hyouka	Eru Chitanda	
Hyouka	Hotaro Oreki	
Hyouka	Satoshi Fukube	
Hyouka	Mayaka Ibara	
Is the Order a Rabbit	Cocoa Hoto	
Is the Order a Rabbit	Chino Kafu	
Is the Order a Rabbit	Rize Tedeza	
Is the Order a Rabbit	Chiya Ujimatsu	
Is the Order a Rabbit	Syaro Kirima	
Sword Art Online	Asuna	
Overwatch	D.va	

Genshin	Keqing	
Vtuber	Kizuna AI	
Rascal Does Not Dream of Bunny Girl Senpai	Sakurajima Mai	
My Teen Romantic Comedy SNAFU	Yuigahama Yui	
My Teen Romantic Comedy SNAFU	Yukinoshita Yukino	

Table 1. The dataset we built

3.2 Construct the dataset

Since anime characters come from different sources, applying different methods fitting with the source can help us to get more high-quality face images, which helps the following training step.

3.2.1 Anime characters

For anime characters, it's better to use the snapshot of the original anime because the shape of the face of that character is usually unchanged in the same anime. Since different painter has a different style to draw pictures, the shape of the face is likely changed. If we use too many pictures like illustrations from the website, it will affect our model's performance.

The summary steps to get the face image for anime characters are listed below. And we used the collection steps on anime KON as the example

$$\begin{aligned} &\textit{Step } 1.\textit{Download animes and store them on our computer} \\ &\textit{Step } 2.\textit{Get snapshots from animes} \\ &\textit{Step } 3.\textit{Recognize anime characters shown in snapshots, cutting snapshots to proper size and label it.} \\ &\textit{Step } 4.\ \textit{Shift images with the same label into the same folder.} \end{aligned} \tag{2}$$

First, we use **lux** [4] and input the webpage of the anime to download the video

```
[→  ~ lux https://www.bilibili.com/bangumi/play/ep21265\?from_spmid\=666.25.episode.0\&from_outer_spmid\=333.337.0.0

 Site:      哔哩哔哩 bilibili.com
 Title:     轻音少女 第一季：第1话 废部！
 Type:      video
 Stream:
     [80-7]  --------------------
     Quality:         高清 1080P avc1.640032
     Size:            358.98 MiB (376419967 Bytes)
     # download with: lux -f 80-7 ...

358.98 MiB / 358.98 MiB [=============================================================================================
Merging video parts into 轻音少女 第一季：第1话 废部！.mp4
```

Figure 3. Using lux to download the first episode of K-ON

Then we use **FFmpeg** [5] to get a snapshot of the video for very 5 seconds. At this time, we store the resulting snapshot. Then we need to select a useful image out of the snapshots. Since some snapshots may contain no anime face image or other useless information, we need to apply face detection to remove the useless part.

```
ffmpeg -i [videoName] -s fps=1/5 %d.png
```

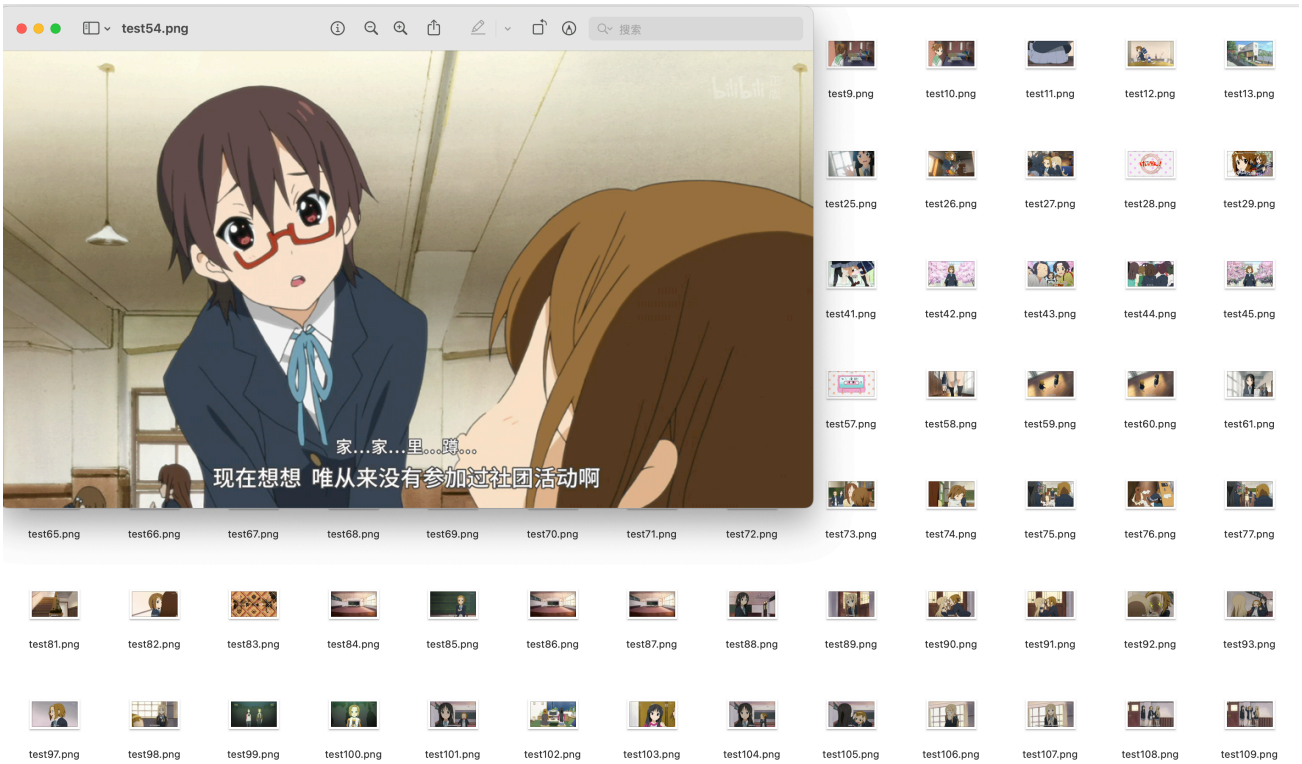


Figure 4. Snapshots we get from the first episode of K-ON

And since the anime characters' faces are quite different from the real-world human face, we need to use one anime face detection. In this paper, I used **anime face 2009** [6] to recognize characters' faces and cut the image.

```
face_collector.rb --src <image dir> --dest <output dir> --threshold 0.7 --margin 0.2
```

We use parameters `--threshold 0.7 --margin 0.2`, and we can get the result like the below picture. The threshold is the critical value, if the possibility of the face is reached this value, the face will be recognized. And the margin is the padding added to each limit of the Axes is the *margin* times the data interval, which is between [0, 1].

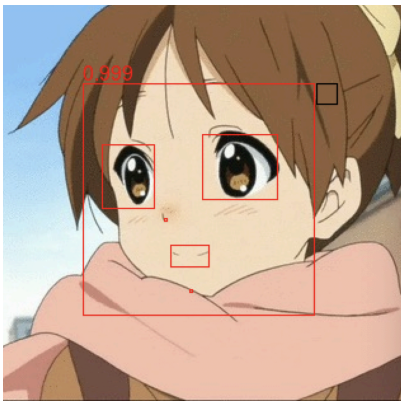


Figure 5. Face recognization examples using Yui Hirasawa's face image from the anime K-ON

After we get the image picture, we can notice that they are sorted by the number of episodes, but to classify each character, we need to sort images by their classes. So we move the character's face image into their classes by hand.

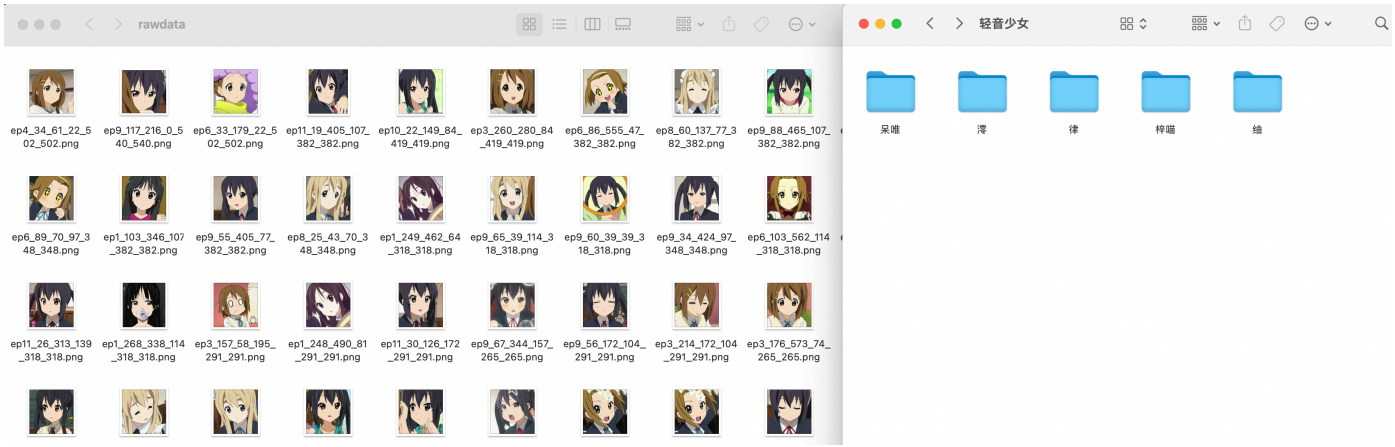


Figure 6. Move face images to folders named by characters' names

3.2.2 Game characters and Vtuber

For characters that come from games and the characters act as Vtuber, we use a different method. Since the characters in the game usually have a different shape than the illustrations. For instance, painters prefer to draw 2D-liked images while games and Vtuber activities prefer to use the 3D-liked image to perform a sense of space. And our goal is mainly to focus on the illustration, so we don't use the original image and tend to use illustrations as the input image. In this paper, the illustrations are all from Pixiv, one of the biggest illustration websites.

Figure 7. The images (left is Keqing and right is Kizuna AI) cut from game and video of vtuber we can find the shape of faces more 3D liked, which will affect our final result

The summary steps to get a dataset from game characters and Vtubers are listed below. Since the steps to get the face images of game characters and Vtuber affects the final dataset. You can find that in the final dataset, the number of face images of game characters and Vtubers is much more smaller than the anime characters. Also, the game characters in this dataset all show up after the final update of the anime face 2009 database, so we need to cut the images by hand.

Step 1. *Use a crawler or manually download the illustrations to our computer.*
Step 2. *Manually get the snapshot of the face.* (3)

We use a crawler called **PixivUtil2**[7] to get the image from Pixiv to get 100 images for one character, so we don't need to move images to different folders this time. As we mentioned before, different painters may have different styles to draw images. So we need to select images with similar shapes of faces by our own hand and remove other images. This step wastes a lot of images actually, it is difficult to find pictures from different painters who have similar styles. Then we need to get square-like snapshots of the face and label the face images.

3.2.3 Reshape and rename images

Then after we get all the result face images, we change the size of them to 96*96, since we need to use the same size input when we apply the random forest algorithm. We also rename the images to keep the folder tidy. Finally, we get the dataset showed before, which contains 21 characters and 3903 images.

4 Algorithm designed

The random forest algorithm is an ensemble learning method for many machine learning fields. It used many decision tresses to train on the same dataset and brings more accuracy than decision trees with a little bias increase. And the Inception V3 model is a convolutional network for assisting in image analysis and object detection, which is the third edition of Google's Inception convolutional Neural Network, allowing deeper networks while using smaller numbers of parameters.

4.1 Inception V3 introdation

The picture below is the network structure of the Inception V3 model.

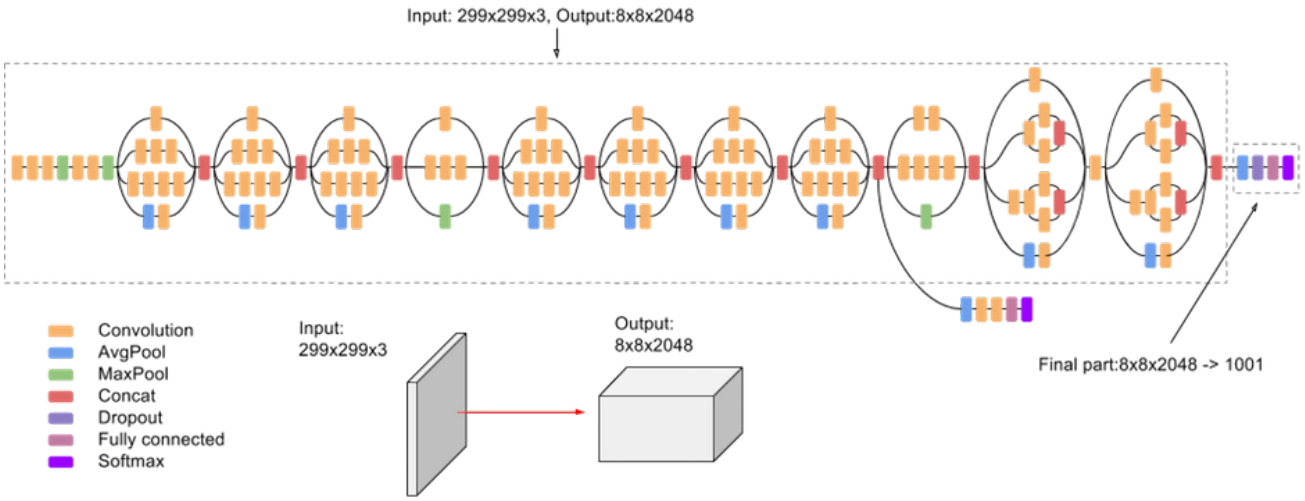


Figure 8. The network structure of Inception V3

The Inception V3 model is a deep learning model based on Convolutional Neural Networks used in classification. It has higher efficiency, a deeper network with the almost same speed, less computationally expensive.

For this paper, we use the model built for ImageNet Competition and apply transfer learning by removing the final part in the picture and retraining the final part to fit our task.

4.2 The improvement from Inception V1 to Inception V3[8]

This improvement comes from 4 modifications, factorization into smaller convolutions, spatial factorization into asymmetric convolutions, the utility of Auxiliary classifiers, and efficient grid size reduction.

4.2.1 Factorization into smaller convolutions

It replaces the 5*5 convolutional layer with two 3×3 convolutional layers, ending up with a net $\frac{9+9}{25} \times reduction\ of\ computation$, resulting in a relative gain of 28% by this factorization.

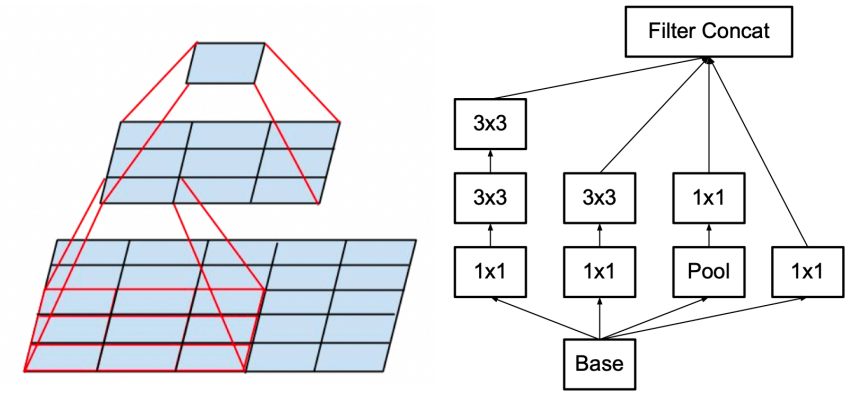


Figure 9. The 5*5 old convolutional layer replace by the new two 3*3 convolutional layers and the new module structure

4.2.2 Spatial Factorization into Asymmetric Convolutions

Then replace the 3×3 convolutions with a 1×3 convolution followed by a 3×1 convolution, and let the two-layer solution is 33% cheaper for the same number of output filters. By applying the two methods, the module becomes like the picture below.

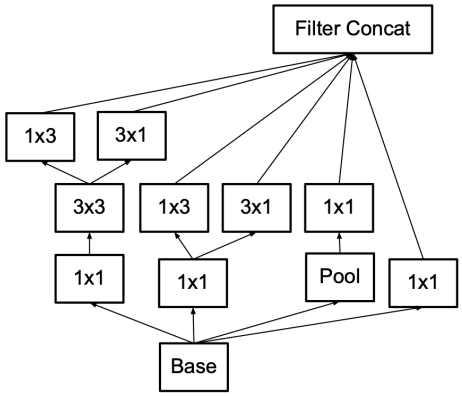


Figure 10. Replace the top two 3*3 convolutional layers to 1×3 convolution layers and 3×1 convolution layers

4.2.3 Utility of Auxiliary classifiers

Also, the auxiliary classifiers are used to improve the convergence of deep networks in this model. Though it won't improve convergence early in the training, near the end of the training, the effect shows up and the model gets more accuracy than the model without an auxiliary branch, resulting in 0.4% top layer accuracy gain.

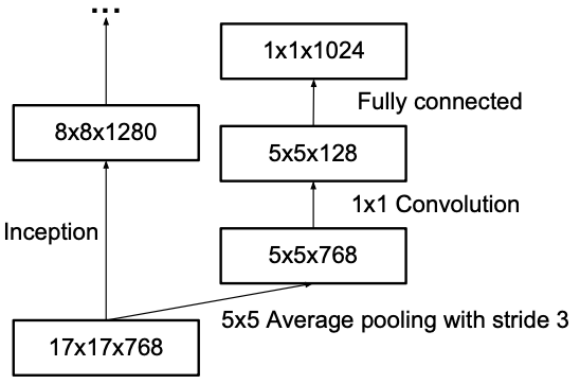


Figure 11. Auxiliary classifiers used in Inception V3

4.2.4 Efficient Grid Size reduction

The activation dimension of the network filters is expanded before the maximum or average pooling to avoid a representational bottleneck. This is done by using two parallel blocks, one is the pooling layer and the other is the convolutional layer to perform it.

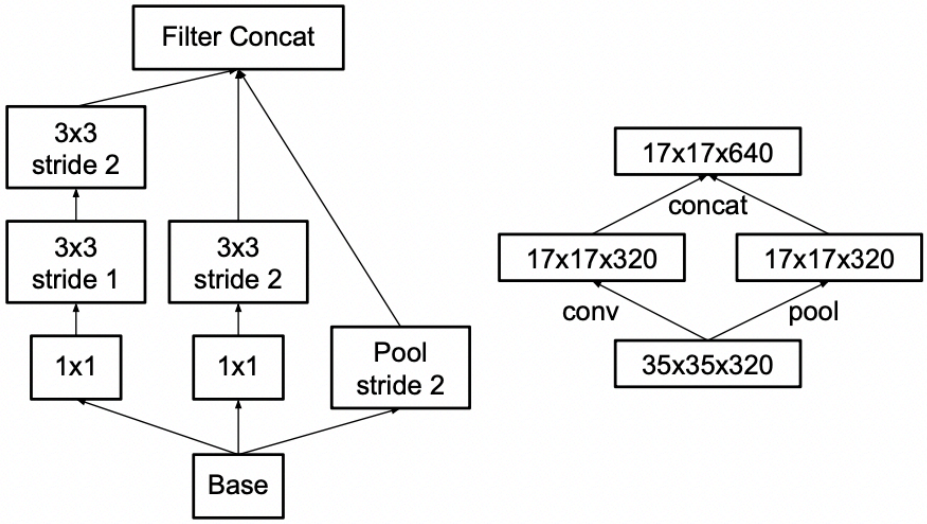


Figure 12. The original module from Inception V3(left) and the module that reduces the grid-size while ex- pands the filter banks.(right)

4.2.5 Model Regularization via Label Smoothing

Label smoothing is a regularization method that introduces noise for the labels making clusters between categories more compact, increasing the distance between classes, reducing intra-class distance, and avoiding the adversarial example of over-high confidence.[9]

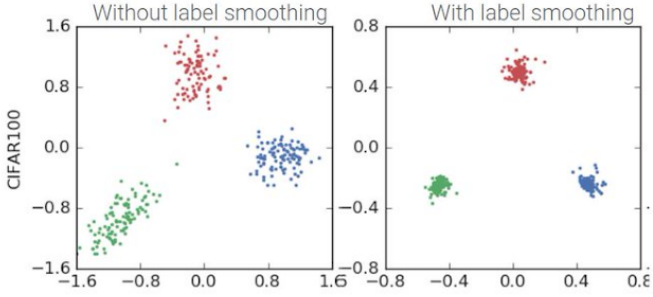


Figure 13. The classification result without(left) and with(right) label smoothing when using CIFAR100 dataset

In Inception V3, they assume a small constant ε and the correct one with probability 1-ε. Label Smoothing regularizes a model based on a softmax with & output values by replacing the hard 0 and 1 classification targets with targets of $\frac{\varepsilon}{k-1}$ and 1-ε respectively.[8]

$$H\left(q', p\right) = -\sum_{k=1}^{K} \log p(k) q'(k) = (1-\epsilon) H(q, p) + \epsilon H(u, p) \tag{4}$$

Thus, LSR is equivalent to replacing a single cross-entropy loss H(q, p) with a pair of such losses H(q, p) and H(u, p). The second loss penalizes the deviation of predicted label distribution p from the prior u, with the relative weight $\frac{\varepsilon}{1-\varepsilon}$. we used $u(k) = \frac{1}{1000}$ and $\varepsilon = 0.1$.[8]

5 Results, Analyses, and Discussion

5.1 Applying random forest

We use the random forest in scikit-learn to apply the random forest algorithm. First, we read the image using `cv2.imread()`, then we use `flatten()` to perform dimensionality reduction in order to store images in one-dimensional arrays. Then we build a DataFrame to store paths, labels, and images. We split both the image and labels into a training set and test set using `train_test_split()` and the test size is 20%. Then we apply the randomForest Classifier to get the result and get the accuracy **0.696629**.

```
rfc = RandomForestClassifier(oob_score=True,random_state=24)
rfc.fit(x_train,y_train)
print(rfc.oob_score_)
print("accuracy:%f"%rfc.oob_score_)
```

5.2 Applying Inception V3

5.2.1 Reason for applying transfer learning

Transfer learning is a machine learning method where we reuse a pre-trained model as the starting point for a model on a new task. Since building a custom deep learning model needs a lot of data and a huge amount of computation resources, in this paper we apply transfer learning. The other reason that we can apply transfer learning in this paper is that our goal of ours and the original model is similar, both are about classification images. And much information used to classify 100 classes in ImageNet is also useful to distinguish the anime characters.[10]

5.2.2 Steps to apply the Inception V3

Download the **retrain.py script**, **label.py script**, and the **Inception V3 pre-trained model**.

Then we can remove the old top layer (the final part of the model) of the model and retrain a new one with our images and labels using **retrain.py**.The retrain.py will train a new bottleneck to recognize specific classes of images. The top layer receives as input a 2048-dimensional vector for each image. A softmax layer is then trained on top of this representation. in our paper, we have 21 labels, so the corresponding parameters are **21** +

2048*21.

Then we modify some of the parameters in the retrain.py, we set the image dir to the folder storing our dataset, the training steps equal to **10000**, the learning rate to **0.01**, **10%** of the images used as a test set, 10% as a validation set and train 10 images as a batch once time and use the entire test set to evaluate the accuracy.

Parameters	Value
training steps	10000
learning rate	0.01
batch size	10
test set proportion	10%
validation set proportion	10%

Table 2. Parts of parameters used in retrain.py script

After executing the retarin.py script, we can get a new output model, a new output label.txt, and retrain log. The log is shown below, the orange line represents the training set, and the blue line represents the validation set. We smooth the line with parameter **0.7**. Finally, we get **87.94%** in the validation set and **99.69%** in the train set. And we mainly focus on accuracy in the validation set. We can find there exist an overfitting problem and we will try to fit it in the improvement part.

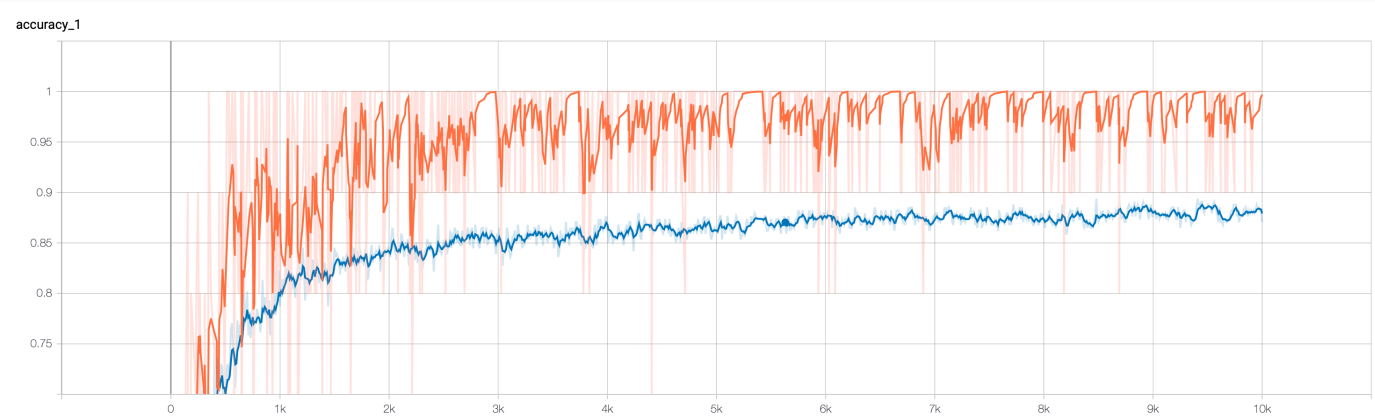


Figure 14. The accuracy of the training set and validation set respectively
The orange line is the accuracy for train set and the blue line is the accuracy for validation set

Compare to the random forest algorithm, we can find that the Inception V3 has more accuracy.

Methods	Accuracy
Random forest	69.66%
Inception V3	87.94%

Table 3. Compare the accuracy of the random forest and Inception V3

After performing retrain.py, we can get a new model called **output_graph.pb** and a txt file called **labels.txt** containing the folder name, which is the labels. After we get the new model, we can use the **label.py** provided to test the results. We change the path of the file, the path of labels, and the path of the model in label.py. Then we download some illustrations about the characters in our dataset to check whether the model correctly recognizes

them.

Azusa Nakano.jpg

Chino Kafu.png

Eru Chitanda.jpg

Hotaro Oreki.jpg

Figure 15. The original test image we used
They are images of Azusa Nakano, Chino Kafu, Eru Chitanda and Hotaro Oreki

And we record the three most likely roles with the possibilities for the label.py script output, and the character name with font bold is the correct result

Image file	Most likely role	Second possible role	Third possible role
Azusa Nakano.jpg	**azusa nakano**0.35730165	keqing 0.22938892	yukinoshita yukino 0.12520486
Chino Kafu.png	asuna 0.33306378	sakurajima mai 0.3093065	keqing 0.19341168
Eru Chitanda.jpg	sakurajima mai 0.3288062	mio akiyama 0.2909892	yukinoshita yukino 0.098570034
Hotaro Oreki.jpg	**hotaro oreki**0.90071654	satoshi fukube 0.054444004	mayaka ibara 0.024812937

Table 4. The output result of label.py script using and images without any cut and Inception V3 we retrain
The results are the character names followed by the possibility
The character name with font bold is the correct result

And we can find that we failed in the middle two images also in the first task the possibility is also low. I guess the reason is that we input images except Hotaru Oreki.jpg are not only images about their face and other unnecessary information that affect the final result. So we manually cut the four images focused on the face and run labels.py again.

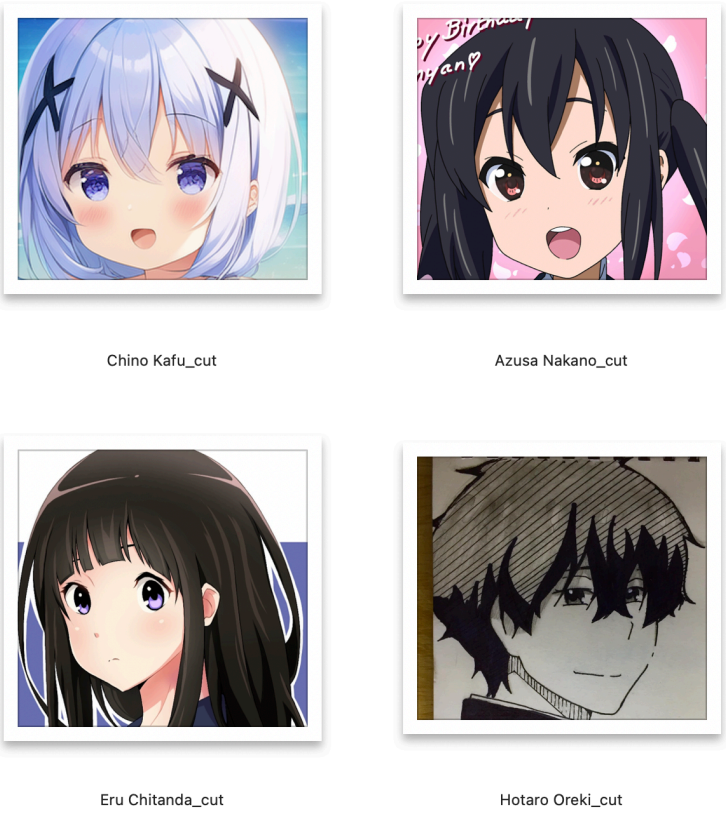


Figure 16. The test image with cut to reduce unnecessary information
They are images of Azusa Nakano, Chino Kafu, Eru Chitanda and Hotaro Oreki

Image file	The most likely role	The second possible role	The third possible role
Azusa Nakano.jpg	**azusa nakano**0.31793657	yukinoshita yukino 0.2763012	asuna 0.15632646
Chino Kafu.png	yuigahama yui 0.5088038	keqing 0.16315687	**chino kafu**0.121952824
Eru Chitanda.jpg	**eru chitanda**0.4583189	kizuna ai 0.22249523	yukinoshita yukino 0.15844545
Hotaro Oreki.jpg	**hotaro oreki**0.92125916	sakurajima mai 0.030491054	satoshi fukube 0.022918124

Table 5. The output result of label.py script using and images with cut and Inception V3 we retrain
The results are the character name followed by the possibility

And we can find that at this time their correct result possibility is all raised except Azusa Nakano. And I can't explain the reason, perhaps it is because our dataset is too small.

5.3 Improvement

5.3.1 Improvement for random forest

For random forests, we can adjust the parameter to improve the accuracy. First, we adjust the **n_estimators** from 10 to 200, and with 10 increasing in each step and find that the best accuracy is appear at 190. Then select 180 as the start point and 200 as the end point to adjust every one increase. Then we adjust the **max_depth** of the tree and find the best value. Then we adjust **min_samples_split**, which is the minimum number of samples required to segment internal nodes, and **min_samples_leaf**, which is the minimum number of samples required to segment internal nodes. Finally, we decide the max_features value, which is the maximum number of features used per tree. We get the best parameters table and after using these parameters, the accuracy of the random forest algorithm reaches **0.7124024**.

Parameter names	Best value
n_estimators	189
max_depth	13
min_samples_split	5
min_samples_leaf	1

Table 6. Best parameters I get

5.3.2 Improvement for Inception V3

We consider image augmentation to larger our input data and try to fix the overfitting problem. Because the important difference between many anime characters is the color of hair and eyes, we did not choose to change the pixels. This data enhancement mainly includes mirror flipping, shearing, shrinking, rotation, affine transformation, all-white or all-black filling, Gaussian blurring, mean blurring, median blurring, sharpening processing, relief effect, changing contrast, moving pixels, distorting the local area of the image. These operations will be randomly applied to the picture. I chose to expand 1 picture to 64 pictures.

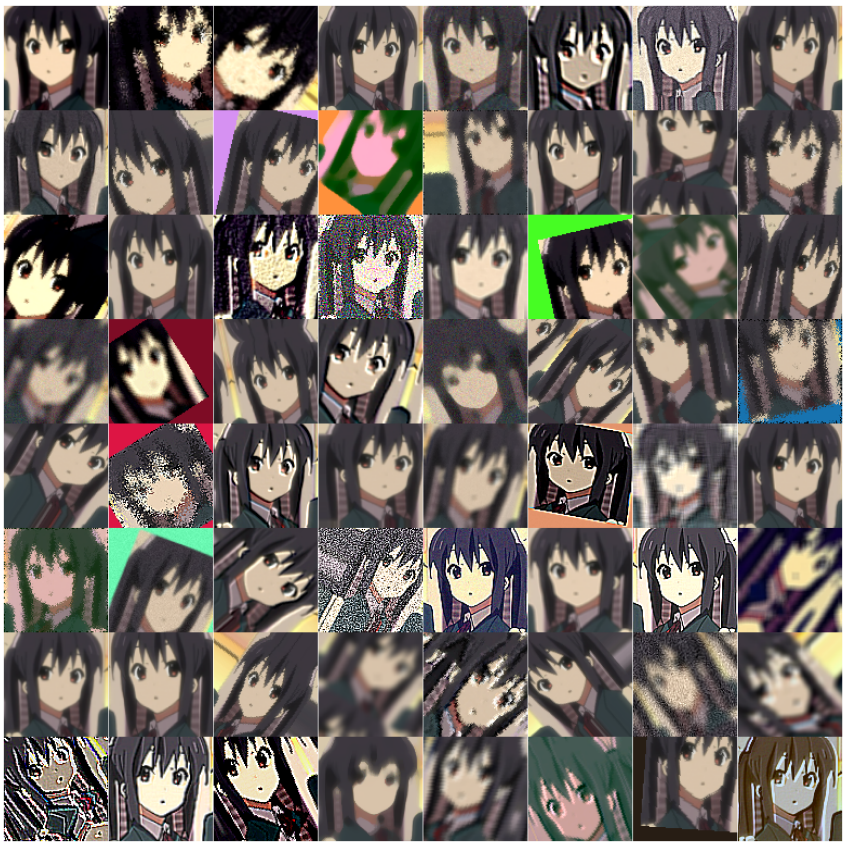


Figure 17. Perform image augmentation to expand 1 picture to 64 pictures

Then we retrain the Inception V3 again using the same parameter shown before. And we reach 89.1% accuracy this time. It can be seen that we have partially solved the overfitting problem, and the accuracy rate of the training set has increased more slowly than before but the overfitting problem still exists.

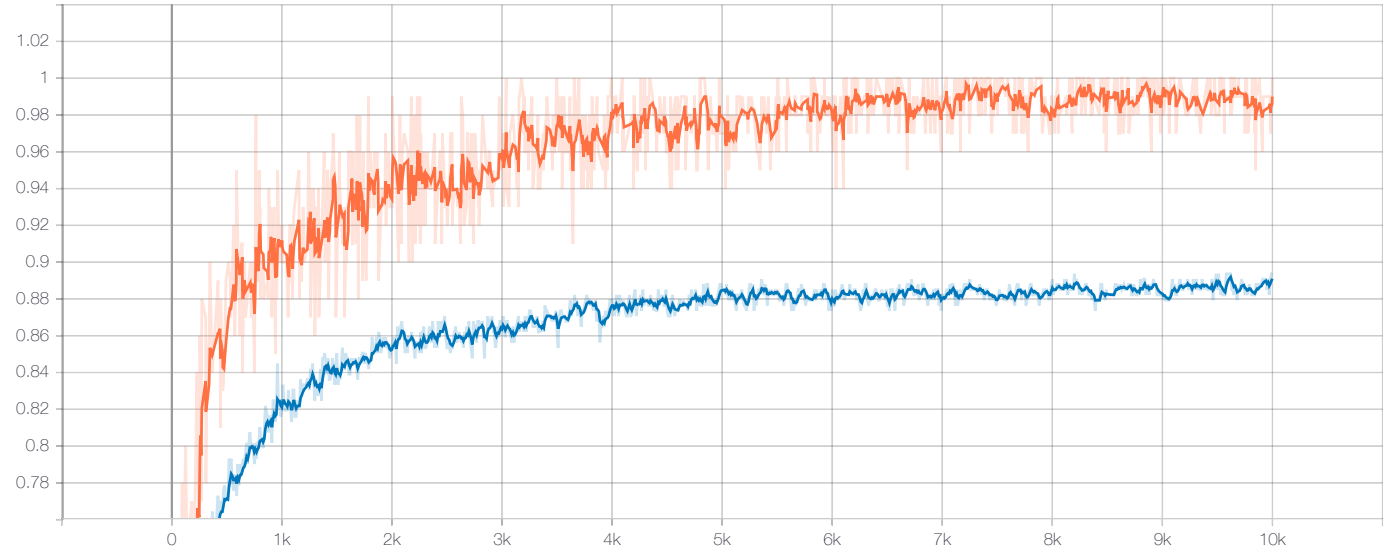


Figure 18. The accuracy of the training set and validation set after performing image augmentation respectively The orange line is the accuracy for train set and the blue line is the accuracy for validation set

Then we compare the accuracy of the random forest and Inception V3 before and after the improvement

Methods	Accuracy
Random forest	69.66%
Inception V3	87.94%
Random forest with adjusting parameters	71.24%
Inception V3 after performing image augmentation	89.1%

Table 7. The accuracy before and after improvement

6 Conclusion

Our goal is to recognize the characters of the given image. In this paper, we first introduce how to get our dataset, it needs to apply more methods such as getting the snapshots and using anime face detection built by others, then we reshape and rename images. For the machine learning part, we apply the random forest algorithms from sk-learn and use transfer learning to retrain the Inception V3 model to recognize the faces of the anime characters. We get 69.66% accuracy by using random forest algorithms and get 87.94% accuracy by using Inception V3. After that, we apply some improvements to increase the accuracy in both the random forest algorithm and the Inception V3 model. We adjust some parameters when using the random forest algorithm and use image augmentation to explain the dataset 64 times to solve the overfitting problems in the original one. Finally, we get 71.24% accuracy by the random forest algorithm and 89.1% by the Inception V3 model after applying the augmentation.

7 Reference

1. Ho, Tin Kam. "Random decision forests." *Proceedings of 3rd international conference on document analysis and recognition*. Vol. 1. IEEE, 1995. https://ieeexplore.ieee.org/abstract/document/598994 ↩

2. Breiman, Leo. "Bagging predictors." *Machine learning* 24.2 (1996): 123-140. https://link.springer.com/article/10.1007/BF00058655 ↩

3. Szegedy, Christian, et al. "Going deeper with convolutions." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015. https://www.cv-foundation.org/openaccess/content_cvpr_2015/html/Szegedy_Going_Deeper_With_2015_CVPR_paper.html ↩

4. Github page of lux https://github.com/iawia002/lux ↩

5. Website of ffmpeg https://ffmpeg.org ↩

6. Github page of animeface-2009 https://github.com/nagadomi/animeface-2009 ↩

7. Github page of PixivUtil2 https://github.com/Nandaka/PixivUtil2 ↩

8. Szegedy, Christian, et al. "Rethinking the inception architecture for computer vision." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016. https://www.cv-foundation.org/openaccess/content_cvpr_2016/html/Szegedy_Rethinking_the_Inception_CVPR_2016_paper.html ↩ ↩ ↩

9. Müller, Rafael, Simon Kornblith, and Geoffrey E. Hinton. "When does label smoothing help?." *Advances in neural information processing systems* 32 (2019). https://proceedings.neurips.cc/paper/2019/hash/f1748d6b0fd9d439f71450117eba2725-Abstract.html ↩

10. Weiss, Karl, Taghi M. Khoshgoftaar, and DingDing Wang. "A survey of transfer learning." *Journal of Big data* 3.1 (2016): 1-40. https://journalofbigdata.springeropen.com/articles/10.1186/s40537-016-0043-6 ↩