GOCHIUSA_FACES, A DATASET FOR ANIME FACES

Rignak

ABSTRACT

We introduce the GochiUsa_Faces dataset, building a dataset of almost 40k picture of faces from nine characters. The resolution range from 26x26 to 987x987 with 356x356 being the median resolution.

We also provide two supplementary datasets: a test set of independent drawings and an additional face dataset for nine minor characters.

Some experiments show the subject on which GochiUsa_Faces could serve as a toy dataset. They include categorization, data compression and conditional generation.

***Index Terms*—** dataset, anime, categorization, conditional generation, data compression

1. INTRODUCTION

1.1. Deep learning and anime-related content

Recently, we saw a growing interest for anime-related deep learning applications. The constant upgrade of hardware enabled to try tasks that could, a decade ago, only be run on low-resolution pictures, for anime world problems. There were tasks ranging from super-resolution to [1] image generation [2] or multi-label estimation [3].

In parallel, there was a dramatic increase in the content of content related to anime through imageboards such as Danbooru or Pixiv. A core feature of such website is a tag system. It allows users (and not only the uploader) to edit the descriptors added to a post. They have the advantage of enabling tagging, giving us a precious source of annotated pictures. Through the years, several datasets were built using these imageboards, such as Nico-Illust [4] contains 400k pictures, or Danbooru2019 [5] with 3.33 m. Still, it is difficult to obtain a large number of pictures for a given character, except for few very popular ones. More so, as they are illustration drawn by a wide range of artists, they have a high intra-class variability and thus are difficult to use for experimentation.

On the other hand, according to the website MyAnimelist, there were respectively 39, 24, 42 and 30 series airing in Winter, Spring, Summer and Fall 2019. With most of them being cours of 12 or 13 episodes, each of them 22 minutes long, and with 24 frames per second, they represent a number of 50 million frames for this single year. This amount, however, is purely fictional as most of the frames are identical. Another issue comes from that it comes with no annotation of the content (except the copyright) when extracted from the videos.

Believing that these frames are a precious source of anime character pictures, we searched for an anime that could have both a limited number of characters (to have a high number of samples for each class) and a focus given on the faces (to assure high resolutions).

1.2. Gochuumon wa Usagi desu ka ?

Gochuumon wa Usagi desu ka ? (abreviated in GochiUsa) is a manga written by *Koi* and published by *Houbunsha* since 2011. As of early 2020, it has been adapted in two cours and two OAV by *White Fox* and *Kinema Citrus*. A third season by *Sentai Filmworks* is expected to be aired in fall 2020.

These anime are good material to fulfil bot our goals.

First, the story is centred around a cast of fewer than ten characters. The screen time allocated for secondary characters is extremely reduced, allowing to have a large number of frames for each main character.

The second interesting point lies in the themes of the original manga. With a focus on the characters, their interactions and the slice-of-life, an important place is given for portraits. This ensures a profusion of high-resolution portraits.

After explaining how the GochiUsa_Faces dataset was build, we will present some statistical data of the pictures and finish with a few use cases for this dataset.

2. BUILDING OF THE DATASET

2.1. Choosing the frames

After getting all Blu-ray files (with a resolution of 1920x1080 pixels), we extracted two kinds of frames: the K-frames and the P-frames.

For compression motives, all the frames of a video are not directly encoded. Two subsequent frequent are, most of the time, very similar. Thus, at a given time, it is interesting to use the previous frames to provide information. These predicted frames are called P-frames. Of course, some of the pictures, usually when there are a lot of dissimilarities, are completely encoded. They are denoted as key frame, or K-frames.

To avoid having too much of identical pictures, we decided to use all the K-frames, but also 1 over 24 P-frame (one P-frame at each second). Almost no K-frame contains characters closing her eyes, so the P-frames are useful to increase the diversity of facial expressions. At the end of the dataset creation, the number of K-frame is 6097 (15.1% of the total) while there are 34420 P-frames.

2.2. Extracting the frames

After getting these dozens of thousands of frames, we used a face detector specifically created for anime faces [6]. Of course, the face detector is not perfect, so we had to remove all the false positives.

Since (i) all frames do not always depict a face and (ii) there could be multiple faces on some frames, it is difficult to predict a number of faces that we could expect beforehand. Thus, it is unknown how many faces were lost *via* true negatives. Simultaneously to removing the false positives, we also removed the faces that didn't belong to either the main classes or the additional classes. We end up 40517 pictures.

As we wanted to avoid any information loss, we did not discriminate the pictures based on their resolutions (except when the face was too small for the character to be recognizable) nor did we resize them. The resolution range from 26x26 pixels to 987x987 pixels. However, the median resolution is 356x356. The figure 1 shows the distribution of the resolution. We choose to plot with a log scale since datasets are often resized to power of two.

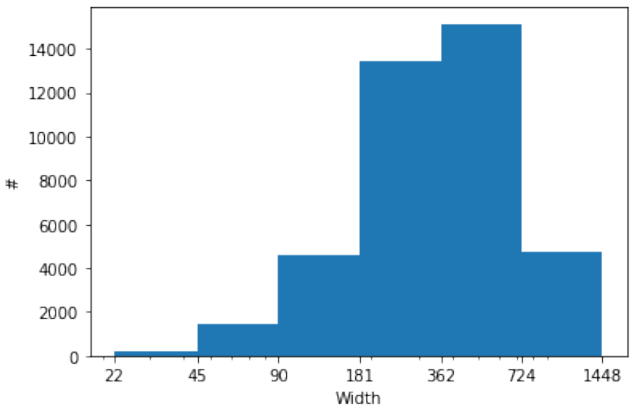


Fig. 1. Distribution of the resolution. The labels on the x-axis are $2^{i+0.5}$ for i from 3 to 11.

All the faces were manually divided into classes, ending the construction of the dataset. All the filenames follow the pattern "{*filename*}_{*frame_type*}-{i}-{j}.png" where :

- *filename* is the name of the video from which we extracted the picture ;
- *frame_type* is either "k" if the frame was a K-frame or "p" if the frame was a P-frame ;
- i is the id of the frame (all K-frames were kept, but P-frames were taken every second so i is to be multiply by 24 to find the real id) ;
- j is the id of the face within the frame (as the face detector could find multiple faces in one frame).

2.3. Training, validation and test set

We do not provide a division between training and validation set. In fact, it is not easy to do so because (i) pictures are heavily time-correlated (especially the p-frames) and (ii) the existence of openings, endings and previews (fortunately, there is no flashback in GochiUsa) imply that some frames are identical through multiple episodes. Thus, it is not possible to randomly divide the dataset in two, nor it is possible to strictly choose the last pictures to make a validation dataset.

However, as we wanted to have independent pictures for testing, we provide a third dataset. Henceforth, we did not extract frames from the anime, but from illustrations scrapped on Danbooru. We only downloaded works with a single character tag, this way we don't have to go through all of them for manual categorization. We applied the same detector as on the frames and through the pictures to remove false positive. For this dataset, the filenames follow the pattern {*id*}_{i}_face.png where *id* is the id of the picture on Danbooru

and *i*, as for the two other datasets, the index of the face within the picture.

You can see samples of each dataset (main, additional and test) in the figure 2.

Fig. 2. From left to right and top to bottom: sample for Blue Mountain, Chino, Chiya, Cocoa, Maya, Megumi, Mocha, Rize, Sharo for (a) the main dataset and (b) the test set, Anko, Chiya's grandmother, Cocoa's mother, Rin, Rize's father, Saki, Takahiro, Tippy and Yura for (c) the additional characters.

The distribution of each class is printed in the figure 3. Sadly, the main dataset is unbalanced with the smallest class (*Blue Mountain*) getting 1067 faces while the *Cocoa* class has 8571 elements. The criterion to belong to the main dataset was to have more than one thousand pictures. However, one could think about avoiding the *Mocha* and the *Blue Mountain* class because of the number of samples, which is not even half of the third smallest.

3. APPLICATIONS

Now that we have introduced the dataset, we will provide few examples of tasks on which it can be used.

3.1. Categorization

The most obvious way to use it is for categorization.

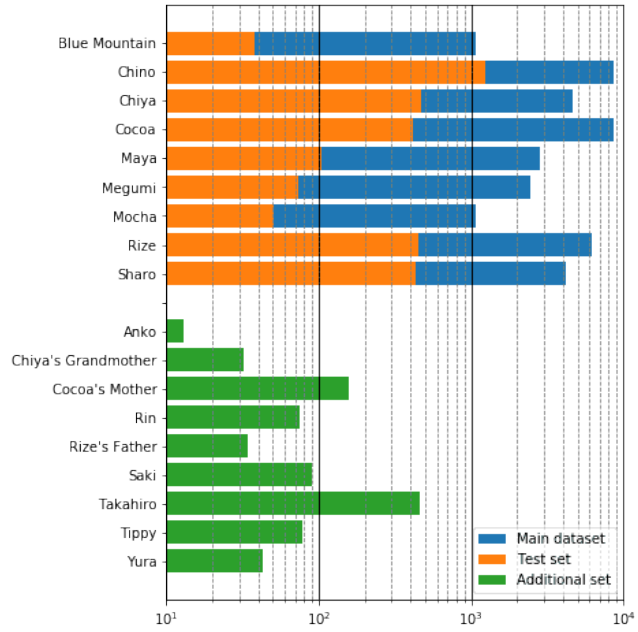


Fig. 3. Number of pictures for each class and each dataset.

In the main dataset, two things can help to discriminate the characters. The first one is the haircut : *Chiya*, Maya and *Sharo* have blunt bangs, *Rize* and *Megumi* are usually depicted with twintails, *Chino* has two x-hair ornaments and *Cocoa* an hairclip. However, as seen in the figure 2, the most recognizable features are the hair and eye colour. There are a few outliers (like sepia pictures taken from the ending of the second OAV), and sometimes the eyes are closed (or blank), but the hair colour still is a good way to make a categorizer.

The figure 4 shows the histograms of the hue component of each class, with 32 bins. They have been corrected by the mean to avoid the peak over the skin colour (coincidentally, there are the same as the maximum values for *Blue Mountain*, *Cocoa* and *Mocha*).

We trained a single layer categorizer taking the thirty-two bins of the histogram as input (thus, we get a very small network with only 330 parameters). With this primitive approach, and no fancy techniques, we get an accuracy of 70% on the validation set, but of only 44% on the test set.

After that, we trained a InceptionV3 to do the same work. On the contrary with the histogram method, the InceptionV3 network needs all the elements of the training set to have the same shape, so we choose to use pictures of size 128x128 pixels (of course, we had to upscale smaller images, and downscale bigger ones). To avoid an upscale too important, we did not include picture smaller than 64x64 in this experiment training set (this is a loss of 1.9% of all pictures). Again without fancy techniques, we got 95% of accuracy on training and 88% on the test set. The confusion matrix of each of the two methods are presented in the figure 5. We can see that the

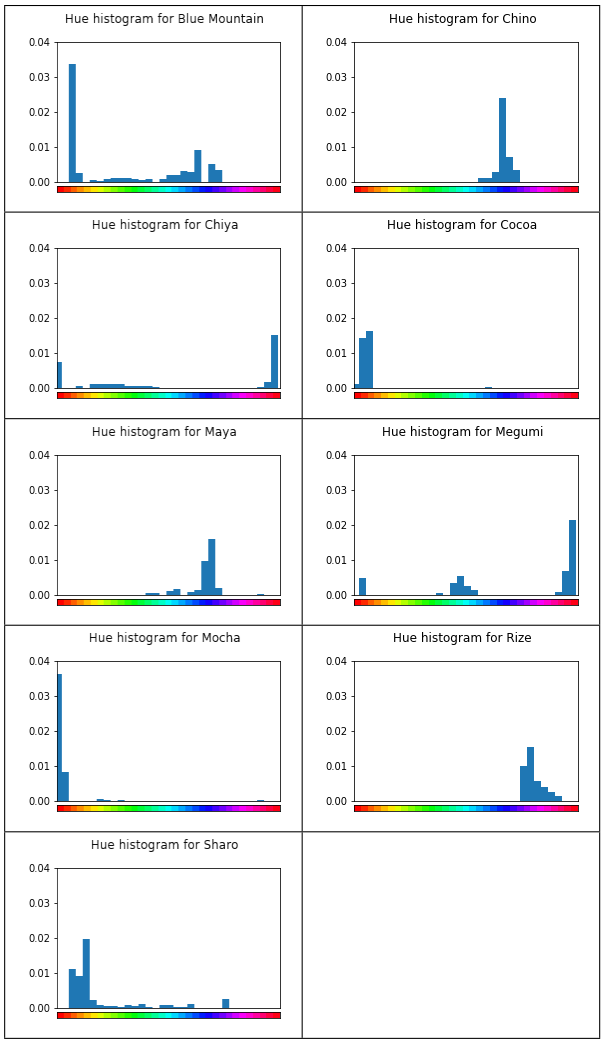


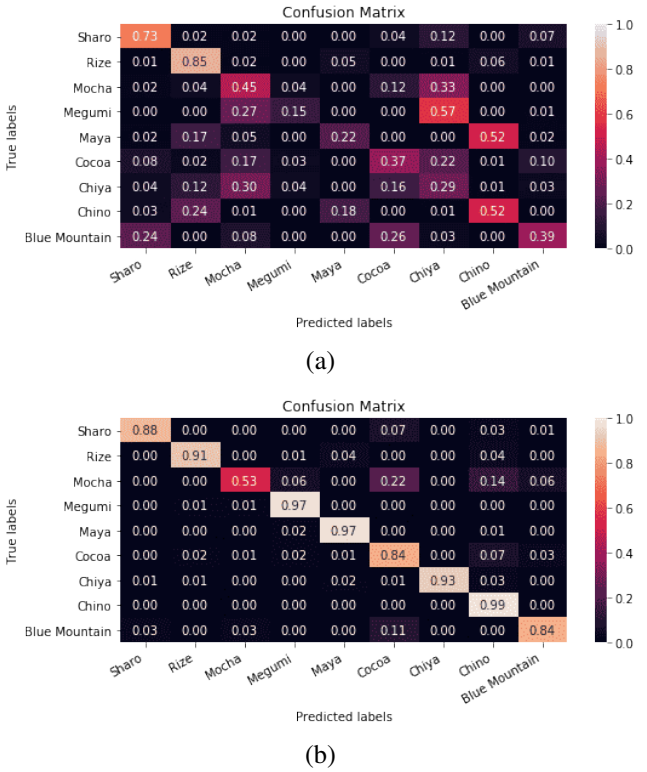
Fig. 4. Difference between the mean hue histogram and the hue histogram of each character.

deep learning model struggle with *Mocha*, having difficulties to segment it from *Cocoa* (as it is her sister in GochiUsa, they share similar hair and eye colours) and Chino (which is more surprising). Overall, this capacity to generalize to the test set, which comes from completely different source, and a much more variability in style, is interesting.

3.2. Data compression through autoencoders

This dataset seems a lot simpler, semantically speaking, than other categorization datasets like CIFAR-10 [7]. Thus, we wanted so see if it was possible to encode them as latent vectors of a very little size.

Thus, we trained a very simple auto-encoder to compress GochiUsa_Faces from 64x64 pixels to a single 32-element vector and did the same with the 32x32 thumbs from CIFAR-10. This auto-encoder is built with convolutional layers of



(a)

(b)

Fig. 5. Confusion matrix of the test set with a single layer classifier using the hue histograms (a) and the InceptionV3 on 128x128 pixels pictures (b).

128 filters, followed by max-pooling to reduce the dimension of the input, until a convolution of 32 filters. The decoder is basically the symmetric.

As we can see on the figure 6.a, most of the pictures from GochiUsa_Face can be encoded/decoded while retaining the main features. The auto-encoder is especially efficient to redraw the eyes. However, it only stands for the ”generic” pictures. For example, profile or blank eyes aren't correctly encoded.

The MNIST dataset is semantically simple, as it can be encoded efficiently on a very small number of elements. On the contrary, the CIFAR-10 dataset, with a very high variety of pictures, is impossible to encode on 32 values.

So we can see that the GochiUsa_Faces dataset could serve as a toy when testing auto-encoders, even if we want to use a resolution higher than the one provided by the MNIST dataset.

3.3. Conditional generation

As we have quite a lot of categorized samples, we have tried a conditional implementation of StyleGAN to generate samples of a given character.

We trained the GAN for twelve hours on a resolution of

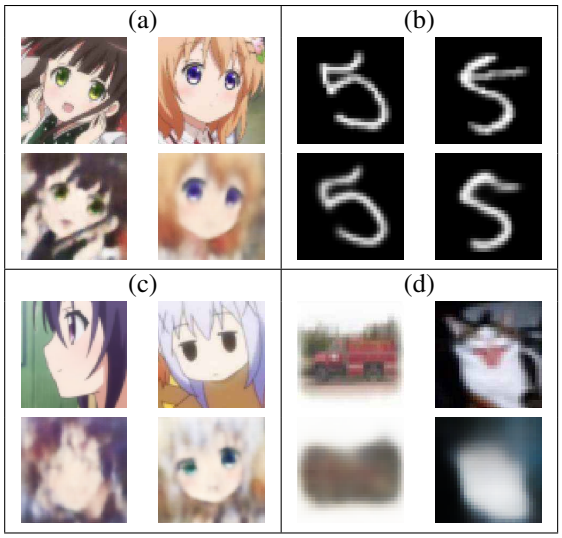


Fig. 6. Truth (first line) and reconstruction (second line) for the GochiUsa_Faces dataset (a & c), MNIST (b) and CIFAR-10 (d).

64x64 pixels on Google Colab. This corresponds to more or less 40000 batches.

The results, though they lack originality and somewhat fuzzy (this could have been solved by longer training, but Google Colab does not support sessions of more than twelve hours, except with the Pro version, not available outside of the USA). Samples for each class are presented in the figure 7.

Fig. 7. Sample generated with a Conditional StyleGAN2

One particularity of the GochiUsa_Faces dataset is that information about the class, as we say with the hue histogram, is mainly stored in the colour space. The figure 8 illustrates, *via* the results of a categorizer with the cGAN outputs, that the conditioning appear very quickly with the GochiUsa_Faces dataset. On the contrary, the classes of the MNIST dataset [8] are discriminated with their shapes. It takes more time to learn how to draw them. The plot figures three trainings for each dataset to minimize the risk of a statistical aberration.

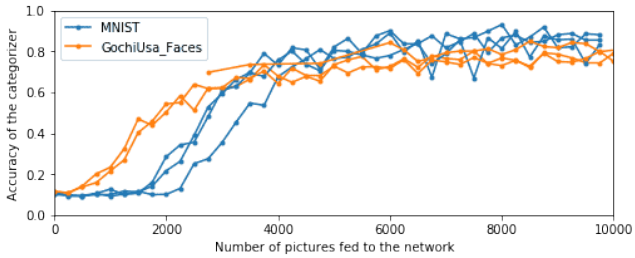


Fig. 8. Performance of a classifier on the generated samples during the training

4. CONCLUSION

We have deployed the GohiUsa_Faces dataset. At the core of this dataset are 39537 pictures from nine characters of the Gochumon wa Usagi desu ka? anime. We also provide a test set of 3262 pictures, from drawing of various artists.

We have shown that this dataset can be used to test a large variety of computer vision techniques, from hue histograms to auto-encoders. It is particularly suitable to test the emergence of conditioning when using conditional image generations.

5. REFERENCES

[1] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang, "Image super-resolution using deep convolutional networks," *CoRR*, vol. abs/1501.00092, 2015.

[2] Gwern Branwen, "This waifu does not exist," https://www.gwern.net/TWDNE, 2019, Accessed: 2020-04-02.

[3] Masaki Saito and Yusuke Matsui, "Illustration2vec: A semantic vector representation of illustrations," 2015.

[4] Hikaru Ikuta, Keisuke Ogaki, and Yuri Odagiri, "Blending texture features from multiple reference images for style transfer," in *SIGGRAPH Asia Technical Briefs*, 2016.

[5] Anonymous, Danbooru community, and Gwern Branwen, "Danbooru2019: A large-scale crowdsourced and tagged anime illustration dataset," https://www.gwern.net/Danbooru2019, January 2020, Accessed: DATE.

[6] nagadomi, "lbpcascade_animeface," https://github.com/nagadomi/lbpcascade_animeface, 2011, Accessed: 2020-04-02.

[7] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton, "Cifar-10 (canadian institute for advanced research)," .

[8] Yann LeCun and Corinna Cortes, "MNIST handwritten digit database," 2010.