Unsupervised Semantic Editing with CycleGAN and Evaluation Framework

Ziqian Liu Pietro Gori* Yunlong He*

*Supervisor: pietro.gori@telecom-paris.fr, yunlong.he@telecom-paris.fr

1. Introduction

The task of learning to separate generative factors in an unsupervised manner has recently attracted a lot of attention in the field of image generation because of its wide applicability to various fields including medical imaging. And most of the existing methods aim to recognize factors in a dataset that contain unique, salient, and semantically meaningful variations (e.g., attributes such as eyeglasses in CelebA), among which the CycleGAN model has attracted much attention due to its **(1)** unsupervised learning, **(2)** end-to-end learning, and **(3)** ability to learn cross-domain representations without the need for a paired dataset. In this paper, we utilize the facial image datasets with and without glasses in the CelebA dataset as a case study to enable the CycleGAN model to automatically identify and decouple shared features (facial structure, etc.) and unique features (glasses) between the two image datasets under unsupervised conditions, and the results are shown in Fig. 1.

Figure 1: Result in the test set (RealA in the upper left,RealB in the upper right of the figure, FakeB generated by CycleGAN model in the lower left, FakeA in the lower right)

In the current task, our goal is to identify significant transformational features in the dataset. As shown in Fig. 1, these features are only present in the target dataset (images with glasses) but not in the background dataset (images without glasses). So both the target dataset and the background dataset should share meaningless (e.g., facial expressions or light changes) variations. In summary, the goal of this research project is to introduce target detection methods to discover additions or modifications in the target dataset compared to the control (or background) dataset as a way of identifying and isolating generative factors common to both populations, the results of which are shown in Figure 2. In addition we were also able to determine the optimal combination of parameters for CycleGAN through the final metrics of target detection.

Figure 2: Fake detection plots in the test set (FakeB in the upper left of the plot, FakeA in the upper right, and the detection results of the YOLO model below)

To achieve this goal, the study employs CycleGAN models incorporating various generator architectures (e.g., ResNet and U-Net) and explores the impact of key hyperparameters (e.g., learning rate and weight

initialization method) on model performance. The CelebA dataset is utilized for training and validation, and the generated results are evaluated using metrics such as Frechette Initiation Distance (FID), accuracy and precision. Moreover, in order to evaluate the generated images more objectively, the study integrates target detection models such as YOLOv8 and GroundingDINO to assist in the judgment, which can minimize the reliance on manual observation.

The main contributions of this research project are: **(1)** Combining the target detection method with the evaluation of the results of the generative model, it provides a new evaluation index method for the generative model. **(2)** Being able to filter the optimal parameter combinations of generative models through evaluation indexes.

2. CycleGAN Model Parameter Tuning

The dataset used in this paper is CelebA(https://www.kaggle.com/datasets/jessicali9530/celeba-dataset). The distribution of images in this dataset covers a wide range of real-world scenarios and conditions, which provides the generalization ability of the model. In this paper, the Eyeglasses feature column in the CSV file provided in the website is used to filter out whether the images wear glasses or not, and then the dataset is pruned, i.e., 13,193 images each of images with and without glasses constitute the dataset used for the subsequent experiments in this paper. Finally, an 8:2 ratio is used to slice the training and validation sets respectively.

All results in the following paper are based on experiments with a Tesla P100 graphics card. The operating system is Ubuntu 20.04, the programming language version is Python 3.8.13, and the deep learning framework versions are Pytorch 1.8.1, Torch 2.4.1, CUDA 12.7.

2.1 Generator Architecture

First of all, there are many hyperparameters in CycleGAN that can be tuned, among them, the generator architecture has "resnet_9blocks, resnet_6blocks, unet_256, unet_128", but because the image resolution is 128*128, so the unet_256 framework is not applicable to this dataset. The specific combination of hyperparameters for training CycleGAN model in this paper is shown in Table 1.

Table 1: CycleGAN's Training Hyperparameter Configuration

Super-Parameter	**Specific configurations**
netD	basic
netG	resnet_9blocks, resnet_6blocks, unet_128
init_type	normal
epoch	400
epoch_decay	400
learning_rate	0.0002
lr_decay_iters	50
beta1	0.5
Batch_size	16
Crop_Size	128

As a rule of thumb, I believe that the image quality generated by the unet_128 model should be the best. To verify this conjecture, after training the model with the parameter combinations shown in Table 1, we were able to obtain the training results of the network architectures with different generators in the dataset, respectively. The visualization results are shown in Figures 3 and 4.

Figure 3: Fake B generated by the highest accurate model trained in 800 epoch

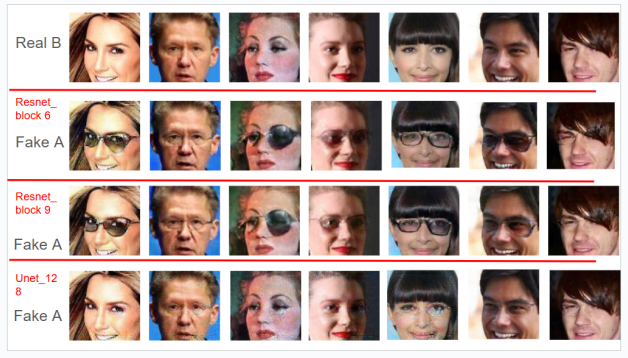

Figure 4: Fake A generated by the highest accurate model trained in 800 epoch

It took me about two days to train the Resnet_6blocks model, about three days to train the Resnet_9blocks model, and about one day to train the Unet_128 model on a P100 graphics card. As you can see from Figures 3 and 4, the Unet model did not successfully transform the eyeglass features and the image is a bit distorted (Mode Collapse). Then I used the FID metric to evaluate the difference between each generated model and the real data distribution. The results are shown in Table 2.

Table 2: The metric of FID

	Fake A	**Fake B**
Resnet_6_block	13.804	13.157
Resnet_9_block	11.249	11.745
Unet_128	63.453	161.658

As can be seen from Table 2, Resnet_6_block has the lowest FID value. resnet_9_block has a slightly lower performance. And the FID value of Unet_128 is much higher than the other two models. This may indicate that Resnet_6_block is more suitable as a generator architecture for the task of this paper. However, in combination with Figures 3 and 4, the Resnet_block model does not perform well in removing glasses and generating glasses, so the model cannot be judged solely on the FID evaluation metrics and the two models will not be studied in depth subsequently. Analyzing the Unet_128 model from a different perspective, the default hyperparameter combination is used, which leads to the Mode Collapse phenomenon of the Unet model and does not fully utilize the generative ability of the model. Based on this, this paper focuses on the Unet model for in-depth discussion and research.

2.2 Hyperparameter Exploration Based on Unet Generators

In that section, this paper only tunes the combination of different learning rates [0.0002, 0.00002] and weight initialization methods [normal, xavier, kaiming, orthogonal] to test the image generation capability of Unet when it is used as a generator. The results of the specific FID evaluation metrics are shown in Table 3.

Table 3: TestSet: FID values for different initialization types and learning rates

Init_type	**Learning rate**	**FID values**	
		testA vs fakeB	**testB vs fakeA**
Normal	0.0002	89.47	48.31
	0.00002	46.20	38.94
Xavier	0.0002	42.94	42.86
	0.00002	46.44	35.08
Kaiming	0.0002	30.20	27.88
	0.00002	94.74	176.85
Orthogonal	0.0002	42.70	32.17
	0.00002	44.86	35.34

To prevent the model from overfitting before having a low FID value in the test set, the FID metrics of the

model in the training set are also calculated here as a reference, as shown in Table 4.

Table 4: TrainSet: FID values for different initialization types and learning rates

Init_type	Learning rate	FID values	
		trainA vs trainfakeB	trainB vs trainfakeA
Normal	0.0002	86.89	45.84
	0.00002	42.20	37.00
Xavier	0.0002	38.27	39.90
	0.00002	42.18	32.49
Kaiming	0.0002	27.42	24.93
	0.00002	91.35	177.99
Orthogonal	0.0002	38.72	30.01
	0.00002	40.57	32.05

Based on the analysis in Tables 3 and 4, we find that the Unet model with a learning rate of 0.0002 and the combination of weights initialized to the Kaiming method trains the model with the lowest FID metrics, and we might have thought that the hyperparameter combination of [lr=0.0002, Init_type='Kaiming'] would be the best. However, when viewed in conjunction with the visualization picture in Figure 5 below, this is not the case.

Figure 5: Some typical output Fake A images of Unet_128 with learning rate 0.0002 and 'Kaiming' initialization types in CycleGAN

As can be seen in Figure 5, at a learning rate of 0.0002 and weights initialized to the Kaiming method. Although its FID value is low enough, CycleGAN does not succeed in learning the features of the glasses and successfully generating or removing the glasses from the image. In addition, in the current field of image generation, it is mainly based on human expert empirical method to observe to evaluate whether the training of the model is qualified or not and combined with the evaluation indexes such as FID to assist in the judgment. This is a difficult problem that needs to be solved.

3. Evaluation Metric for Image Generation Models

To address the hard problem raised in the second point of this report, the object detection model is combined in this section as an evaluation metric for the image generation model. For the task of this paper, our core idea is that we need to use the model to detect the face and glasses in the RealA, and to detect only the face and not the glasses in the corresponding FakeB. In contrast, use the model to detect only the face in RealB and to detect both the face and the glasses in the corresponding FakeA. With the above judgment conditions, we can determine whether the CycleGAN model learns the features of glasses under unsupervised learning conditions. So we need to design two models, one for detecting faces and the other for detecting glasses.

For face detection, I found many pre-trained YOLO models of detecting faces. After going through the experiments, I found that this model gave the best predictions (https://github.com/Yusepp/YOLOv8-Face). So I do not train the model of detecting face, and based on pre-trained model to test the CelebA dataset. The visualization is shown in Figure 6.

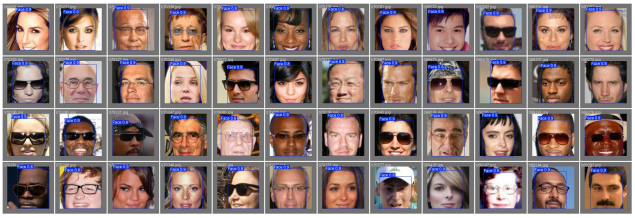

Figure 6: Visualization test results of YOLO_Face pretrained models on the dataset

Furthermore, on the test process, I use "GroundingDINO" to detect the face and output the format

of YOLO labels (I designed a util. Because Groundingdino predicts the coordinates of the rectangle of the output glasses in the graph, and then I visualize the predicted rectangle based on the output coordinates). The precision of this open source face detection model tested in the CelebA dataset is shown in Figure 7.

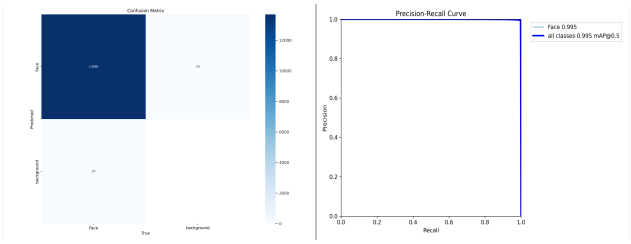

Figure 7: Metrics for YOLO_Face model on the testset

Combined with Fig. 6 and Fig. 7, the model for face detection performs well on the CelebA dataset, so for face detection, this paper is based on this open source model for experimentation. According to the same way, I also tried to find a pre-training model for eyeglasses, but very unfortunately did not find a relatively perfect one (or the open-source pre-training model didn't detect well on this paper's dataset). So next we need to work on spectacle detection; the experimented methods in this paper are shown in Table 5.

Table 5: Introduction of method

Method	Description
OpenCV (haarcascade_eye_tree_eyeglasses.xml)	Traditional Visual Inspection Methods
Glasses_detector	Open source model specialized in detecting eyeglasses
GroundingDINO	A Multimodal Large Model for Guiding Target Detection Tasks via Input Textual Prompts
YOLOv8	Target detection model with high performance

3.1 OpenCV and GlassDetector

OpenCV is an open source computer vision and machine learning library, and we use the cascade classifier (haarcascade_eye_tree_eyeglasses.xml) provided by it for face detection based on pre-trained Haar features from traditional methods. In addition, we use Glasses_Detector for face detection. It is a deep learning-based eyeglasses detector that utilizes the CNN network to extract and classify the features of eyeglasses. I experimented with 50 images from each of the photos with and without glasses in the CelebA dataset, and some of the visualization results obtained are shown in Figures 8 and 9, respectively (The blue matrix is generated from the previous YOLO_face prediction, which is then fed into the respective eyeglasses detection model and finally predicted to generate the following red or green matrix.).

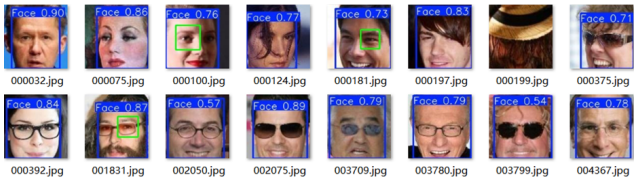

Figure 8: Visualization test results of OpenCV on the dataset

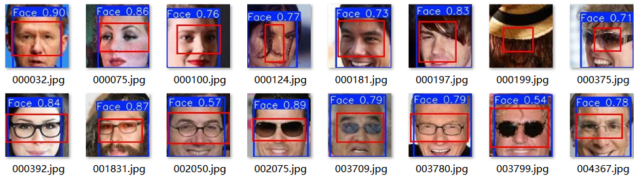

Figure 9: Visualization test results of Glasses_Detector on the dataset

As can be seen in Figures 8 and 9, OpenCV and Glasses_Detector did not perform well, with a large number of misdetections and omissions, so these two methods were excluded.

3.2 GroundingDINO and YOLO

The intention here is to use GroundingDINO to label the spectacle targets in the CelebA dataset and then generate labels in YOLO format to train the YOLO model. However, I found that it doesn't label well (GroundingDino models may be mislabeled, i.e., the full image is labeled as the label of the eyeglasses), and the visualization is shown in the Figure 10, so that the YOLO model trained to detect eyeglasses will not predict well.

Figure 10: The prediction label of GroundingDINO (The red line crosses out the detection error, i.e., the entire image is labeled as a spectacle target)

But don't worry, by looking at Fig. 10, we find that all the wrongly detected images are those where the original image is not wearing glasses, but GroundingDINO detects them as wearing glasses and treats the whole image as a target. So in this paper, we write a filtering logic to determine whether it is a full image frame by checking whether the dimensions (bbox_width and bbox_height) of the labeled boxes in the YOLO label file generated by the Groundingdino model are both greater than 0.95. If this holds, the labeling of that image is set to 0 (i.e., no glasses are worn).

3.2.1 Training of the YOLOv8 Spectacle Detection Model

By the above method, currently we have obtained the YOLO format txt file of the CelebA dataset, which is the category of each image and the bounding box coordinates of its eyeglasses target (if eyeglasses are present in the image), respectively. The combination of hyperparameters for training the Yolo model is shown in Table 6.

Table 6: Training hyperparameter combinations

HyperParameter	Specific configurations
imgsz	128
batch	32
lr0	0.01
lrf	0.01
weight_decay	0.0005
momentum	0.937
optimizer	Adam

The YOLOv8 model was trained to detect eyeglasses based on the combination of key hyperparameters as shown in Table 6 (the rest of the parameters not mentioned are default parameters). In order to get the best performing model for the metrics, this paper was retrained from scratch based on the YOLOv8x model architecture and also fine-tuned based on the YOLOv8x pre-trained model provided on the official website. The final performance metrics on the test set are visualized in Figures 11 and 12.

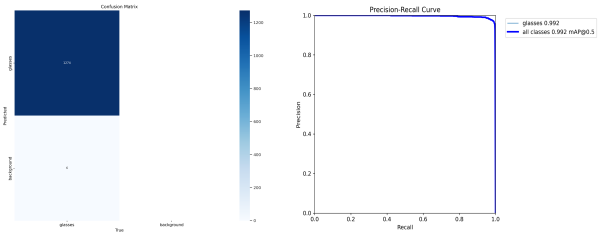

Figure 11: Metrics for Fine-Tuned model on the testset

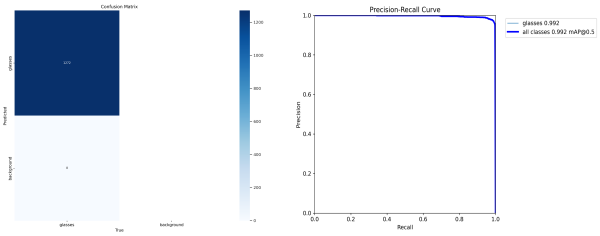

Figure 12: Metrics for Re-Train model on the testset

As can be seen from the figure 11 and 12, the performance metrics of the Fine-tune model on the test set (a total of 2000 images) are approximately the same as the retrained model. Overall, the fine-tuned model is slightly better than the model trained from scratch. So subsequent experiments are based on the fine-tuned model here.

4. Evaluation Indicators Based on Object Detection

Until now, we have the trained eyeglasses detection model and the pre-trained face detection model. Based on the logic rules mentioned in the previous subsection

3, this paper conducts experiments using these two target detection models. Examples of successful and unsuccessful removal of glasses from the FakeB generated by the CycleGAN model based on the RealA are listed, and examples of successful and unsuccessful addition of glasses from the FakeA set generated based on the RealB set are also listed. A partial visualization is shown in Figure 13.

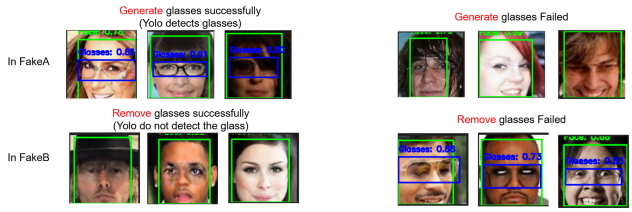

Figure 13: Examples of judgment results (the picture in the FakeA collection is valid if it is detected for both glasses and face. the picture in the FakeB collection is valid if it is detected only for face)

4.1 Evaluation Metrics

Firstly, I filter the original CelebA based on the trained face and glasses YOLO model, i.e., if the images in the original dataset of trainA and testA cannot be detected by both glasses and face, they are directly deleted. images that cannot be detected by face in trainB and testB are also directly deleted. The visualisation results as shown in the Figure 14.

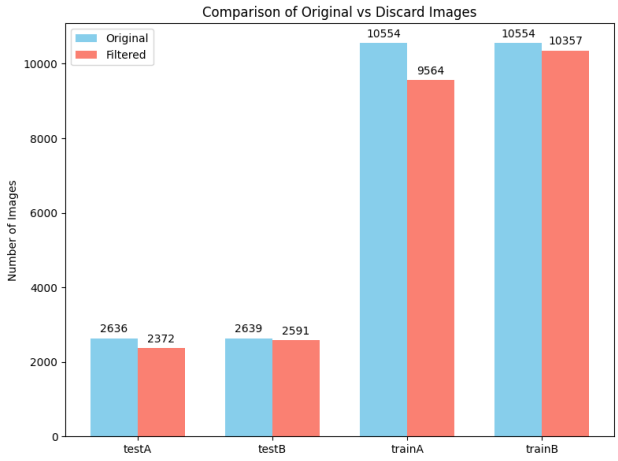

Figure 14: Comparison of Original vs Discard Images

As shown in the Figure 14 are the visualization results, the filtered results(images) are subsequently used for confusion matrix calculations etc. In addition, this paper continues the experiments here along the hyperparameter combinations of different learning rates [0.0002, 0.00002] and weight initialization methods [normal, xavier, kaiming, orthogonal] in the second point of the above article. That is, to address the difficulties raised in the previous section, the aim is to try to replace human observation and reduce human workload with a target detection model. The visualization metrics for each parameter combination are shown in Fig. 15, Fig. 16, Fig. 17 and Fig. 18.

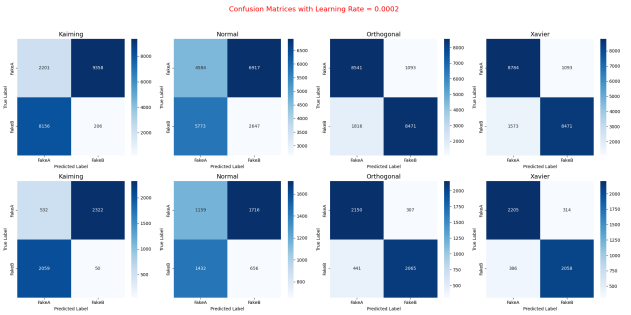

Figure 15: Confusion matrix with a learning rate of 0.0002 and different weight initialization methods (The top graph is the training set metrics and the bottom graph is the test set metrics)

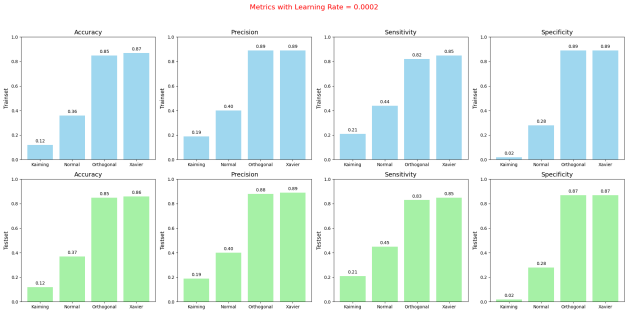

Figure 16: Metric with a learning rate of 0.0002 and different weight initialization methods (The top graph is the training set metrics and the bottom graph is the test set metrics)

Figure 17: Confusion matrix with a learning rate of 0.00002 and different weight initialization methods (The top graph is the training set metrics and the bottom graph is the test set metrics)

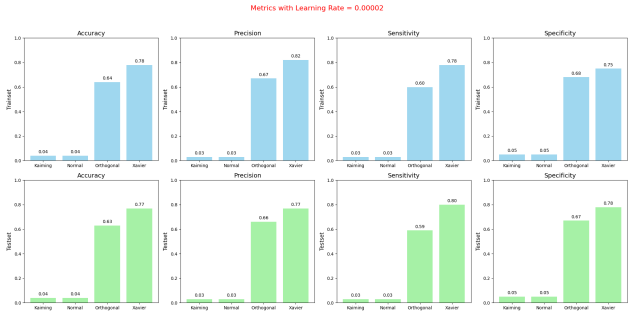

Figure 18: Metric with a learning rate of 0.00002 and different weight initialization methods (The top graph is the training set metrics and the bottom graph is the test set metrics)

With the four figures presented above, we can analyze the data from several perspectives using the control variable method. First, examining the learning rate (Figs. 18 and 19), at a learning rate of 0.0002, we observe that the Orthogonal and Xavier weight initialization methods yield superior results. This conclusion is based on the fact that both the training and test sets achieve accuracy and precision metrics ranging from 85% to 90%. In contrast, the Kaiming weight initialization method underperforms across all metrics, suggesting that the CycleGAN model fails to learn the features associated with glasses, despite its low Fréchet Inception Distance (FID) metrics. The validation results further support the findings illustrated in Figure 5. Therefore, at the same learning rate, we can deduce that the Orthogonal and Xavier weight initialization methods are preferable for model training. Similarly, we can quantitatively analyze the performance at a learning rate of 0.00002. Additionally, we observe that the Orthogonal and Xavier weight initialization methods, which performed well at lr = 0.0002, show a decline in evaluation metrics at lr = 0.00002. Thus, we can conclude that the model is better trained at a learning rate of 0.0002 when using the same weight initialization methods.

If you are not confident that such a combination of parameters is optimal, you can continue to adjust the learning rate for experiments to narrow the interval of the optimal learning rate. Here the experiment was continued with lr=0.001 and lr=0.0005, and the results of the metrics are shown in Figures 19 ,20, 21 and 22.

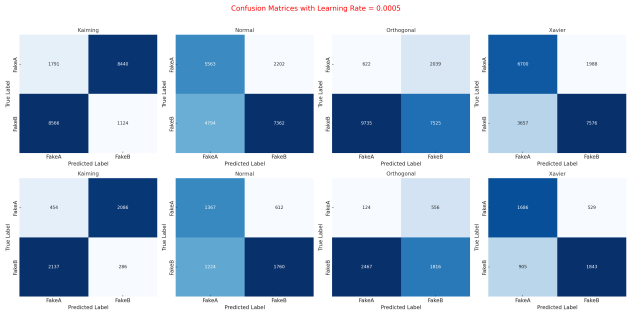

Figure 19: Confusion matrix with a learning rate of 0.0005 and different weight initialization methods (The top graph is the training set metrics and the bottom graph is the test set metrics)

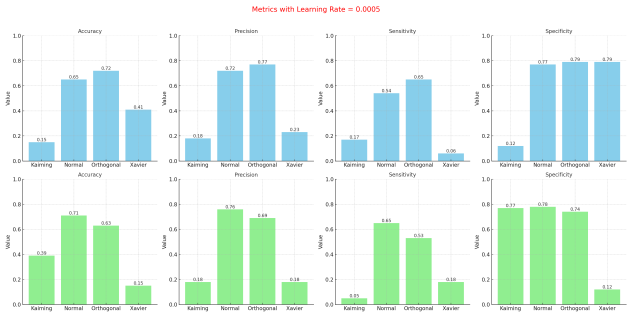

Figure 20: Metric with a learning rate of 0.0005 and different weight initialization methods (The top graph is the training set metrics and the bottom graph is the test set metrics)

Figure 21: Confusion matrix with a learning rate of 0.001 and different weight initialization methods (The top graph is the training set metrics and the bottom graph is the test set metrics)

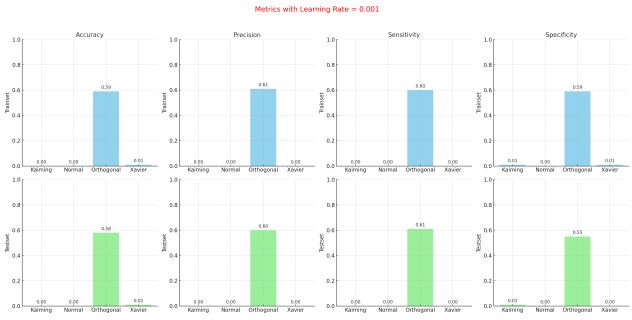

Figure 22: Metric with a learning rate of 0.001 and different weight initialization methods (The top graph is the training set metrics and the bottom graph is the test set metrics)

In summary, the optimal parameter combination for the model applicable to the dataset presented in this paper is a learning rate of 0.0002 and an initialization type of 'Xavier'. Additionally, we can further investigate the performance of various initialization types on metrics such as accuracy and precision at learning rates of 0.005 and 0.002. This will help us identify a more accurate and suitable learning rate for this model.

5. Conclusions

This study explores various generator architectures and hyperparameter configurations using CycleGAN models, while also proposing a novel evaluation approach for image generation. The focus of this article is on the performance of generating or removing spectacle features within the CelebA dataset. The results indicate that the ResNet-based model struggles to accurately transform the target spectacle features, despite achieving low FID scores. In contrast, the U-Net architecture demonstrates greater potential but is susceptible to mode collapse under the default hyperparameter settings, highlighting the necessity for more targeted parameter tuning. In this paper, we experiment with different combinations of learning rates and weight initialization methods, ultimately finding that the combination of Xavier initialization and a learning rate of 0.0002 yields the best results in terms of accuracy and precision metrics for this dataset.

However, low FID scores do not directly correlate with a model's ability to effectively capture meaningful features, underscoring the limitations of metrics like FID as evaluation tools for image generation models. Furthermore, the reliance on empirical judgments made by experts indicates the necessity of incorporating additional evaluation methods based on target detection. This approach would allow for a more comprehensive assessment of the models' success in learning and transforming specific features.

CycleGAN model's learning rate (lr) and initialization type (Init_type) hyperparameters can indeed be further optimized. However, this paper primarily aims to present new ideas and will not delve into repetitive details. The central concept is that to establish more robust parameter configurations for the U-Net model, we should experiment with a wider range of learning rates, such as 0.0005 or 0.001, given that lr=0.0002 has already demonstrated effective performance. By gradually identifying the parameter combinations that best suit the network architecture and dataset, we can enhance model performance. Additionally, integrating advanced multimodal models like GroundingDINO will unlock new opportunities for optimizing specific feature generation tasks. Automating the evaluation process through target detection models will not only alleviate manual workload but also provide a more objective framework for assessing generative models.

Future research could concentrate on improving the synergy between CycleGAN models and target detection frameworks to achieve more accurate feature transformation. In summary, this study offers valuable insights into optimizing the CycleGAN architecture and emphasizes the significance of developing comprehensive evaluation methods tailored to specific tasks.