Generating Anime Character Faces Using ACGANs

Mrinal Walia
School of Computer Science
University of Windsor
Windsor, Canada
SID: 110066886

Aayushi Navinchandra Patel
School of Computer Science
University of Windsor
Windsor, Canada
SID: 110087817

Dhruvkumar Arvind Patel
School of Computer Science
University of Windsor
Windsor, Canada
SID: 110055817

Aaditya Pradipbhai Parekh
School of Computer Science
University of Windsor
Windsor, Canada
SID: 110084734

***Abstract*-**This paper will implement a baseline Auxiliary Classifier Generative Adversarial Network capable of generating new anime character faces. We aim to design a style-guided discriminator and a generator network and train it on 63,000 synthesized anime faces. ACGANs can be notoriously hard to train as the generator, and the discriminator architectures are susceptible to parameters, hyperparameters, regularization, learning rate, and activation functions. However, unlike a conventional GANs network, our proposed framework uses a GPU for faster training and can improve its performance given more c-omputational power and datasets. This problem has previously been solved but lacks a generic architecture. Hence, our proposed framework can be used as a baseline network across different applications of GANs. The generated anime faces from our final model are visually pleasing and resemble designer-generated characters. As generative modelling is an unsupervised task and does not expect images to have labels, the dataset used in this project has 63,000 unlabeled cropped anime faces available at this URL: https://github.com/bchao1/Anime-Face-Dataset.

Abbreviations: Auxiliary Classifier Generative Adversarial Networks (ACGANs), Generative Adversarial Networks (GANs), Artificial Intelligence (A.I.), Machine Learning (ML), Convolutional Neural Network (CNN)

I. INTRODUCTION

Generative adversarial networks (GANs) are a primer for developing a custom neural network to learn to yield new, synthetic examples of data that can pass for actual data. There are two neural networks, Generator and Discriminator, pitting against each other. You can feed the network a sample of random noise as input, and they can contest with each other to evolve more precisely in their projections. In this era of high computational power and advanced innovations, GANs were introduced **[1]** by Ian Goodfellow and his fellow mates at a research lab at the University of Montreal. GANs are considered applications in almost any field creativity is an essential element. From image, voice, and video generation to supreme research in biomedical research, GANs have proved their potential by providing innovative solutions. As GANs can mimic to learn from any type of data, it is believed to have opened gates for many good and evil characters. However, in this paper, we will see the bright side of GANs and how they can be used to achieve incredible creativity in animation.

Due to the fabulous results shown and generated by GANs in the past decade, one of the studies that we will examine in this paper has previously been considered a bit complex due to the requirement of high computational power and the amount of dataset required to achieve the highest accuracy possible. This paper is focused on Generating anime character faces using ACGANs, which stands for Auxiliary Classifier Generative Adversarial Networks. Like most true artists, three French students utilized the power of GANs by borrowing some open-source code to put the first A.I. painting in Christie's auction and were able to sell the portrait for $ 432K **[2].** This abstract level of accuracy and precision to learn intricate features from millions of data points is made possible by the two different types of architectures used in the contest to figure out things and generate valuable data even from random noise. In this network, the job of the generator network is to create new and realistic-

looking anime face images, while the discriminator's job will be to distinguish between the authentic images and the fake images generated by the generator network in conjunction. In simple words, the generator network tries to fool the discriminator model. Once the generator can defeat the discriminator and achieve the desired accuracy, the generative model is used to create new plausible anime face samples on demand.

Although GANs have particular use cases mainly depending on the different domains, it is widely used to interpolate new faces, generate high-quality images, and generate face generation **[3]**. We have chosen anime characters' faces because of the limitations of the datasets of natural persons and the lack of computational resources. This paper will start with understanding the problem statement by defining the motivations and justifications for the proposed algorithm and experiments. Then we will review the background to understand the methodology more clearly. We will also show the computational experiments with relevant images and samples to prove our architecture.

II. PROBLEM STATEMENT

This section will discuss the problem statement to describe and formulate our techniques. Also, we will discuss the motivation that led us to select this project and then justify the chosen approach.

A. PROBLEM DEFINITION

A.I. is booming in every domain. The realism it is in solving various problems in Image Processing is outstanding. However, the evaluation of photo realism of AI-synthesized faces indicates a gap in the algorithms needed to achieve the desired goals in the desired computational time and simplicity. As a part of the study, face generation in high-dimensional space to match the facial features extracted from the sample of images in the dataset is complex. Of course, many researchers and a team of participants are trying to solve this problem. Still, scientists say there is more worrisome that it might toss suspicion on real photos and videos. The course we are taking to develop the discriminator and generator architecture generates stunning actual anime faces at first glance. Still, they do not exist as they were born from the mind of a computer. The technology used is ACGANs, developed by a group of researchers from google brain and initially released for relation extraction. These high-quality relational faces from ACGANs are used to improve the performance of relational faces over the natural source and help generate meaningful experimental results **[4].** that prove that our proposed architecture design significantly improves the performance of relation extraction compared to other state-of-the-art methods.

B. MOTIVATIONS

The motivation to take this method of specialized GANs and work in image synthesis is the computational advancements in A.I. According to the recent theory suggestions, the accuracy is affected by the datasets and the approach used to define the problem. To examine the possibility, we examined results from different research papers with different approaches to understand the current state of the face generation problem. We concluded that there are many possibilities to be taken care of, and from the application side, it has endless options. There is also recent shreds of evidence to suggest that the anime character faces generated by the computer can be realistically used in children's magazines and animated shows which saves the multimedia and mass-communications companies a lot of money and resources from hiring designers for their teams, who else, would spend hours of brainstorming sessions and resources to come with a new character. In contrast, on the other hand, the A.I. anime face generator model can give you thousands of options to choose from in seconds or minutes.

C. JUSTIFICATIONS

Face generation has been considered the most incredible idea in A.I. technology in the last 20 years. The best method behind the news of artistic style transfer, face-swapping, natural voice generation, music synthesis, innovative image generation, etc., is a kind of A.I. called a GAN. The

task of interpolating new faces from an existing dataset has proved to be achieved with state-of-the-art results from the different types of the GANs architectures. **[5]. [6]. [7].** shows the different approaches adopted for research in time to generate faces from a synthetic dataset. It also proved that a conventional neural network can be fooled into misclassifying the faces and features by adding only a tiny amount of noise into the sample or training dataset, which is insufficient for this problem. Also, GANs architecture is designed to learn the probability distribution of a dataset by competing two neural networks against each other, which helps to have a double feedback loop with the ground truth of the images we have given to the network. Hence, the GANs architecture can generate persuasive samples from almost any distribution with proper time and dataset. They are entertaining and perplexing as it is a technology that takes us one step closer to the world where we depend more and more on the power of computers and A.I.

III. BACKGROUND STUDY AND RELATED WORKS

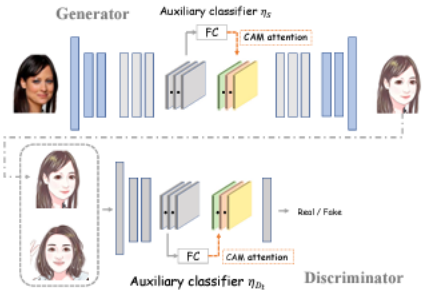
This section will briefly discuss the face generation problem's background. Usually, people do not understand how neural networks and deep learning is impacting our daily lives. When we see the results of the work, it is when we become aware of what is happening around us in the technological space. Therefore, the generation of fake faces using artificial intelligence and machine learning has become a growing concern for every tech news magazine in the late 2020s. Not everyone could guess the faces generated by the state-of-the-art GAN network **[8].** The results are astonishing to see how well technology has evolved in the last five years. The fake portraits generated look very realistic, and it is frightening how A.I. can generate faces for itself and look so accurate as a human face. In the history of artificial intelligence, this problem has been solved before using a conventional neural network which was not very close to a GAN network. As the neural network was getting good at predictions of unseen statistics, it was a good approach and had recently become good at engaging in dialogue. However, the face-to-face conversion model **[9].** Aimed to read and generate facial features and gestures alongside fine details, allowing the model to adapt its outputs based on the expressions of the face. However, deep convolutional neural networks (DCNNs) **[10].** Yet another first-class algorithm has tried to achieve generalized face recognition across viewpoints and facial expressions. The face representations in DCNNs are compact and, with feature units that are not turned, retain exact details about face images in addition to the original face identities. The DCNNs follow a semantic interpretation of representing faces in sparse trajectories in the 3D space instead of being interpretable by the facial features unit activations. The proposed algorithm in **[10].** Is a generalized face generation algorithm and a hallmark of human face generation for familiar faces and to ground them in the broader context of previous-generation face generation algorithms. It shows a new class of visual representation that allows for face generation and representation but does not assure generalized face generations.

There were other experiments done using GANs in **[11].** To generate fine-grained cartoon faces for multiple categories of sufficient training data while the given new group samples extend the basic model by creating a basic translation model for the entire group. Group-specific branches are constantly changing to generate high-quality cartoon faces for various groups of images. As seen in the image **[11.1].** The sample uses the generative adversarial network to learn the mapping functions from paired training images, and paired data is tough to get hold of. Thus, CycleGANs tries to learn translation from two mapping functions simultaneously and achieve outstanding results, as shown below.

[11.1] FIGURE OF GAN NETWORK MANIPULATING FACES

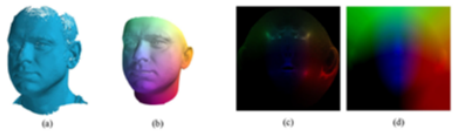
The overhead outcomes from the paper's few shots of knowledge transfer for fine-grained cartoon face generation concentrated on face-to-cartoon transformation. Still, their practice very much aligns with our network design. Therefore, it is needed to regard such distinctions to acquire on the internet for adequate performance. At the same time, anime face generation of specific styles is hard to collect or design, and the amounts of variable data of each group are also unbalanced. GANs have presented a lot of awareness since the first paper, stating that face generation is feasible in operating this network and in training generators and discriminators.

In distinction, the second network attempts to determine the generated model, whereas the preceding model tries to understand from each iteration some new elements. The paper aimed to learn n mappings between two categories where one denotes the natural face, and the other denotes the cartoon faces. For comfort, it is directed to the groups like 0 and 1, making the discriminator's job a binary classification task. Thus, it can be preferably trained to utilize a mapping network operating back propagation algorithm. Whereas in the second training loop, the multi-branch network architecture learns fine-grained anime face generation using a few shots. Below is the basic model architecture presented in **[11].**



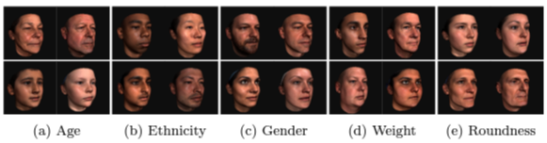
[11.2] FIGURE OF GAN ARCHITECTURE

In late 2020, generating realistic 3D faces using a trunk-branch GAN was made possible by **[12].** They showed how a 3D face generation revolves around linear statistical models and how GANs can be used for generating high-quality texture shape information. Also, it was shown how a specific state of the generation on the presentation and the simultaneous samples can be in the additional documents. 3D face modelling and remodeling extensively utilize current progress in generative modelling. The correlation between the geometry and texture a 3D face model can synthesize the 2D faces randomly for face generation without constraints. However, to feed the shape and the surface for the normal of the facial meshes into a deep convolutional network for reshaping them into an image-like tensor format as seen in the image **[12.1].** They represent a raw mesh in the registered using the large-scale face model template and the interpolated 2D U.V. map.



[12.1] FIGURE OF GENERATED FACE HEATMAP

Furthermore, we see in detail the modification of GANs as conditional GANs to generate the 3D faces with expressions like the MeIn3D dataset. It included 35,000 facial scans of a special exhibition with a 3dMD face capturing all subjects registered. The sample dataset also had 46% male and 54% female faces. The resulting expressions had the sad, neutral, happy, angry, etc., expressions of faces, covering 7 of the universal expressions. Below is the variation of generated 3D faces by different face categories, and readers are recommended to zoom in on a digital version below.



[12.2] FIGURE OF DIFFERENT FACIAL INDEPENDENT FEATURES

The above model is highly generalized and has different categories to distinguish the new face. In some of those categories, the users are encouraged that our expressions generator can synthesize quite a diverse set of expressions. Similarly, the users can see more word

generations in the different generations of the six universal expressions. A set of benefits of such unified 3D face modelling shows that the correlation among texture by PCA models is missing photorealism and an articulated illustrated with the extended head representation of their system. In conclusion, we wanted to show how neural networks and other GAN architectures have grown with time and how the previous work proposed by experts helped us achieve our project's results.

IV. METHODOLOGY

This section will discuss where the material and data were collected. Also, we will plunge into the deep of the suggested ACGAN network and the requirements and beliefs. Lastly, we will see some of the formal complexities we faced while doing this project.

A. MATERIAL AND DATASET

The anime face dataset we have employed is publicly open at Kaggle at the URL: https://www.kaggle.com/datasets/splcher/animefacedataset. The dataset contained more than 63K images of high quality but remember not the real faces, only the anime faces. And one thing to notice was that all the anime characters were girls, and when we go into the folder, we have an image folder. The folder contains 63632 images in high-quality scraped from the website getchu, and then the images were cropped using the algorithm bcpcascasde. The size varies from 90*90 to 120*120 pixels for each image, so you can rescale them before using them. The images have clean backgrounds and rich colors with few outliers like bad cropping results or non-human faces. The images are in the /src/images/ folder, while the cropped images will be in the /src/cropped/ folder. Some samples from the used dataset of images are shown below:

Fig1. Samples from the training dataset

B. PROPOSED METHOD

The approach we have suggested to work on generating realistic images of anime characters' situations is the class of GAN named Auxiliary Classifier GANs with regular training **[13].** And softmax cross-entropy loss. It is widely known that training an ACGAN is not easy. Our project presents the gradient exploding in the classifier and casting input vectors into a team of the hypersphere. The Reagan benefits of the state-of-the-art algorithms are the model weights and a software package that provides implementations of representing cans and all experiments in our paper. Below can be seen the two generators and the discriminator model architecture in detail as we have implemented.

DISCRIMINATOR NETWORK

The discriminator network is simply a classifier that takes an image as input and tries to distinguish accurate data from the data generated by the generator network. In this style, it could use any neural network architecture depending upon the type of data it is classifying. We can use a CNN that gives a single number output for every input image. This network requires tighter integration between the generator and the discriminator. Hence, we use a stride of 2 to progressively reduce the size of the output feature maps in the training process.

GENERATOR NETWORK

The generator network is a part of GAN architecture that learns to create fake anime faces with the help of a discriminator network and by using a vector of the matrix of random numbers, which is used as a source for generating an image.

This network learns to discriminate and classify its output as real or fake by converting a latent tensor of shape (128, 1, 1) into an image tensor of shape (3, 28, 28). In short, the generator generates new anime characters' faces that are as close and like the anime character faces as provided in the real dataset. We will use the ConvTranspose2d layer from the PyTorch framework to achieve this. This performs a transposed convolution, also called deconvolution. The generator network will be outputting an image as shown below and this image will be given as input to the discriminator network to return if it's real or fake.

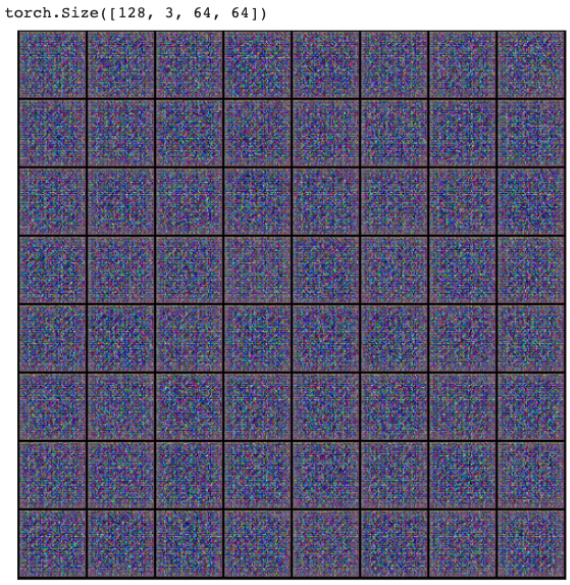


Fig2. Output of the Generator Network

C. CONDITIONS AND ASSUMPTIONS

So, in our proposed algorithm, we have made some hypotheses and assumptions based on the type of networks. First, the discriminator network is considered a binary classification model. We will use the binary cross-entropy loss function to quantify how well the model can differentiate among the generated anima character faces and the original images from the dataset. Below we have explained the main points that are a part of the training of the discriminator network in our proposed methods:

- The discriminator network gives us a prediction of 1 if the predicted image was selected from the real Anime Face dataset, and it will give us a 0 if the image was generated using the generator part of the network.
- We pass a batch of natural images to compute the loss and set of target labels to 1
- We are also passing a different batch of fake images generated using the generator directly into the discriminator network, and this is done to compute the loss, and we are also setting the target labels to 0
- Finally, we add both losses and apply them to the overall loss value to calculate the gradient descent and adjust the discriminator network weights.

When coming to the generator network, it gives images as output. Hence, we take a different approach to using the discriminator as a part of the loss function for the generator training process. Below is the step-by-step approach is taken for training the generator:

- The generator network gives us a batch of generated face images to pass into the discriminator network.
- We calculate the loss by setting the target labels to 1, meaning natural, to fool the generator's network.
- We are using a loss function to perform the gradient descent, which is the change in the generator network weights. It gets better at generating actual type face images to fool the discriminator network by reaching the end of the training epochs.

D. FORMAL COMPLEXITIES

We operate the fit method from the neural network library PyTorch that trains the discriminator network and generator network in resemblance to individually other for each batch of the provided input training data. For optimizing the loss, we are using a binary cross-entropy loss function with Adam Optimizer as an activation

function with some additional custom parameters that are well known for the discriminator-generator network. The below image represents how we are giving a random input vector to the generator model, and it generates examples for us that is then along with real examples given to the discriminator model to make the final binary classification for real or fake prediction.

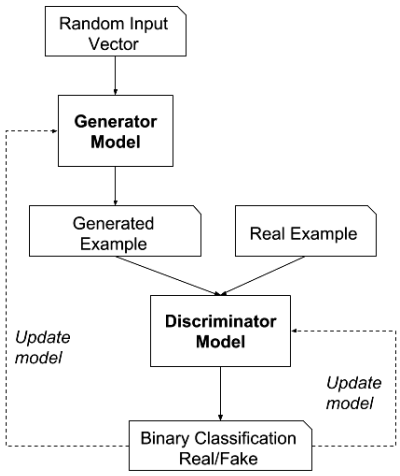


FIGURE OF FULL NETWORK IN TRAINING

V. COMPUTATIONAL EXPERIMENTS

This section will discuss the experiments implemented to train the GAN network, evaluation metrics plotted to validate the model and comprehensive particulars of the proposed method, and results. Then, we will end this section with a discussion on the computational experiments.

A. EXPERIMENTS

The network we have designed in the project is a complex GAN network with two main components: A generator and a Discriminator. After seeing the design of the architecture in the previous section, we tuned the parameters and hyperparameters to change the model's performance and validate the results. We have tried different parameters and hyperparameters. However, the best values that suit our model can be seen in the code at the URL. We are now going to discuss the different values that played a significant role in experimenting with the ACGAN network. Before that, we are saving the outputs from the final full training model to check the results at every epoch, the output of which can be seen below:

```
Clipping input data to the valid range for imshow with RGB data ([0.
Saving the images.... Generated_Images-0000.png
```

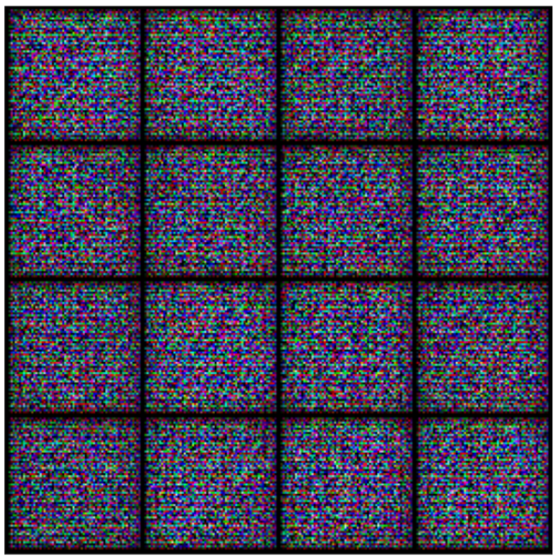


Fig 3. Image after every epoch

LEARNING RATE

In the field of A.I., a learning rate is a tuning parameter that is used within an optimization algorithm. It helps us determine the step size at every iteration in the network. It gradually moves towards the minimum of a loss function in the gradient descent plot. When we changed the learning rate value in our experiment from 0.0001 to 0.001, we saw that the models' weights were getting updated during training, and the lower the learning rate was, the better the model was performing, and the loss was getting reduced. So, we decided to settle upon a value of 0.0002 by doing some data visualization of the loss values of the generator and the discriminator network.

EPOCH

In the field of A.I., epochs are the time taken for the one complete pass of the training dataset through the final network. It is an essential hyperparameter for the GAN network, as it helps determine the times the entire training dataset will be passed through the network to learn the

features of the anime character faces. We trained our network for 100 epochs as there was a constraint on power and time. We decided to stop the training process at 100 epoch values for both factors and the optimal performance of the model.

STRIDE

Stride in machine learning and image processing is a component of neural networks that helps to tune the compression of the image dataset. We are using a stride value of 2 that will output a matrix of 2x2 across every pixel value of the image in training set at every discriminator network layer.

PADDING

Padding in machine learning is another experimental hyperparameter that allows us to add an extra number of pixels to an image when it is being processed by the kernel of the network. We have decided to add extra padding of 1 at every corner because the images in the dataset were of external dimensions and have much variability in the values of every pixel as every point represents a part of the anime character's face.

BATCH NORMALIZATION

Batch normalization is a training parameter and method used to normalize the output of the hidden layers and speed up the training process. We have decided to add a batch normalization layer to each of our hidden layers in both networks with a value of 512.

ACTIVATION FUNCTIONS

The activation function is a hyper parameter used to adjust the outputs of every layer in the network by applying some algorithm in the background depending on the algorithm used. We have decided to use LeakyReLU, which is short for Rectified Linear Unit. It is a type of activation function with a slightly negative value instead of a flat value from it preceded ReLU.

KERNEL SIZE

We have used a kernel of size 4x4 in the input vector as the dimension of the image vector in the first layer was 64x64. As the first layer has lesser input channels and is less critical in size, we settled on the value of kernel as 4x4.

B. EVALUATION METRICS

We evaluate the model after every 10-epoch value by saving the model's output prediction locally. We have used the log metric method of the Jovian library in Python and machine learning. It allows us to save the loss of generator and discriminator and the actual and fake scores for the model. The plot can be seen below. We are also plotting the charts to examine the loss and accuracy of the model. For this, we used OpenCV and matplot library, which helped us better understand how the model was performing when changing the values of the different parameters and hyper parameters in our network. After performing the experiments and implementations, we also use the commit method of the jovian library to commit the state of the model in real-time and save the model's checkpoints and weights from being used for evaluations in future.

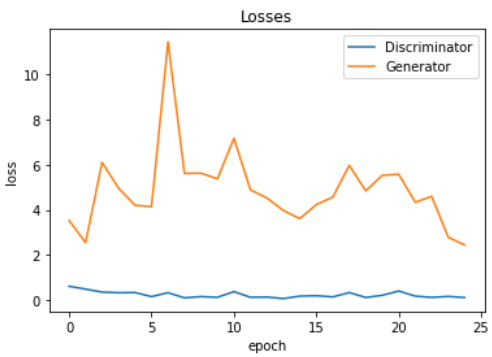
C. IMPLEMENTATION

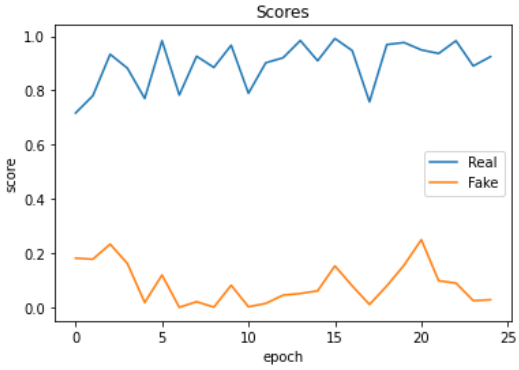
We have used many online platforms and tools to implement the project and the deep learning and neural network coursework research. The first is Google Collab, an IDE for machine learning tasks that runs in the cloud-hosted by google. It is easy to use and the best IDE for performing intense deep learning tasks in the cloud with a significant amount of computational power from google. For programming the code, we have used Python, the best language for GANs implementation and easy to use and offers a wide variety of libraries and methods built to be used. Next, we used the PyTorch framework to design and train the network. It is a machine learning framework that accelerates the training process with its object-oriented structure. We have also used some utility python libraries such as scikit learn for the math’s and algorithms behind the GAN network,

matplotlib for visualizing the results and evaluations, and OpenCV for loading and manipulating the image vectors. Other tools are GitHub for saving the code and keeping track of the project's progress. The training could further be improved by training the network on more than 500 and less than 1000 epochs. Also, one thing to notice is that, as the ACGAN network are complex architectures networks, we are using GPU to train the model. If you have a Windsor or Linux O.S., then you can use CUDA GPU by Nvidia to train the network locally on GPU. Or the other option is like ours, where we are using a jupyter notebook by Google to train the network on GPU at a much higher speed.

D. RESULTS

As we can see from the plots, the loss of the discriminator network at 25 epochs is at a value of close to ~ 0.2, and the loss for the generator network is at ~ 2. And from another figure, which is for the accuracy score, it can be seen that the authentic images have an accuracy of close to ~ 97%. In contrast, the ACGAN network gives a precision on the fake dataset of ~ 12%. Now what we are observing from the above image, we can say that the network could perform much better if trained at 500-1000 epochs. This is where the limitation was. It needed high computing and processing power to achieve this task in the desired time. Now, talking about the accuracy and loss of the generator and discriminator network, we can say that the model is trained exceptionally well and has shown satisfactory performance at this smaller number of epochs. It shows that our model is complex yet has excellent outcomes. The final evaluation of the face images at each epoch can be seen in the next section.





E. DISCUSSIONS

In the discussion of our computational experiment, we achieved an accuracy of 97%, and the model's accuracy can be stretched up to 99.8% with more computing, processing power, and resources. We will show the incremental results, showing the generated output of anime face characters from every five epochs until 25 epochs. When we compare the results, it is observed clearly how the generator network is generating better real-anime character faces with every epoch and how the discriminator is getting fooled at understanding the almost real-like generated faces images.

Epoch-1

Epoch-5

Epoch-25

Epoch-10

Epoch-20

VI. CHALLENGES/ LIMITATIONS

As GANs are highly tender to data used, and as our project has an enormous amount of image data, it was the first challenge we faced to decide what architecture to use and what will be best in terms of accuracy, time, and cost complexity. So, after doing comprehensive research and multiple brainstorming sessions, we came across ACGAN architecture, which was the best for our defined use case. It had a discriminator and a generator network. The core task of each is to generate fake data through learning the potential probability distribution, and the discriminator should output the sample and send the sample to the discriminator. Despite the algorithm's better accuracy, we decided to go with our own experience and hit-n-try method to tweak some parameters and hyper-parameters to see what we could come across. To solve our problem, solutions have been shared above sections, and to address these challenges, several previously discussed solutions as given in **[14].** Also, the insights gained and the promising research results were shared in our report, and in this analysis, we performed a sweeping survey of the architecture. Another challenge we faced was knowledge sharing, as all the team members had their working schedules and styles of explaining things. We, at even intervals, used to have scrum meetings and better understand each participant's concepts. To handle the abovementioned issue, we used an alternative training methodology: we used generated images and real examples in the

discriminator to get the results and calculate the loss. We compared the target with the loss to calculate the accuracy. We train the discriminator and then the generator, and then we proceed.

VII. CONCLUSION

In this section, we will conclude our findings and summarize the report, the possible future research options in this area, and what potential problems can be solved in the coming times.

A. SUMMARY

In this report, we have discussed the anime character face generation using a computational approach to make the new data. The motive behind this project was to help the business, communication and tech industries explore the prospects of the GANs architecture, which is comparatively new in this generation of computers. We proposed a network of neural networks in the discriminator and the generator algorithm. Our proposed technique can outperform the results of some of the papers already proposed in this area. We have proved our results with the loss and accuracy plots by showing the newly generated images by the GAN network. The outputs are much better than any other paper in the market. We have used a deep neural network at the core, mainly used for unsupervised learning like generative modelling. Machine learning involves automatically discovering the learning qualities and patterns in the given dataset. This way, the model can generate new output or examples that have never been drawn from the inputs. GANs could be notoriously difficult to train; hence we decided to use the GPU version of the google colab, and those with a GPU can use the CUDA version of the NVidia graphic card.

B. FUTURE RESEARCH

Although GAN networks have improved in the last few years, many areas remain to work on. Current progress from simple low-resolution images to high-resolution images in the predicted outcome can be achieved. Here, we are looking for the images generated by the generator model at every epoch should be of very high quality such that the discriminator fails in fewer iterations and will save us a lot of time. But however, there is a trade-off to this. The data's complexity increases, affecting the training time as we don't have enough compute power to process high-quality image data at every epoch in the least amount of time possible. The cutting-edge augmented reality feature is another future aspect to improve and work on. How can the generated faces be more realistic and in 3-dimension? This can be achieved by working on the A.R. feature of the GAN modelling. The 3D face metrics obtained from the GAN network can then be used with the A.R. compiler like opacus or unity to provide a more realistic feel to the application.

C. OPEN PROBLEMS

The open problems to work on in this project can be finding the trade-off between using different GANs and other generative models and what sort of distributions these different models generate. This will also help to scale the GANs beyond image synthesis and can be further expanded to say about the global convergence of the training dynamics, how we evaluate the GANs and what we should use to scale training with batch size. One more point to add here is the relationship between GANs and their adversarial examples that can further be explored.

VIII. ACKNOWLEDGEMENT

We would like to expand our appreciation to Dr. Ngom for his immense support during the project and for inspiring us to choose Anime Face Generation using ACGANs for our project.

The project code and all the necessary files to run the project (including the project demo) are accessible on GitHub:
https://github.com/abhiwalia15/Generating-Anime-Character-Faces-Using-ACGANs

REFERENCES

[1]. https://arxiv.org/abs/1406.2661

[2].
https://www.theverge.com/2018/10/23/18013190/ai-art-portrait-auction-christies-belamy-obvious-robbie-barrat-gans

[3].www.sciencedirect.com/science/article/abs/pii/S026288562100024X

[4]. An Auxiliary Classifier Generative Adversarial Framework for Relation Extraction

[5]. StyleHEAT: One-Shot High-Resolution Editable Talking Face Generation via Pre-trained StyleGAN

[6]. Complete Face Recovery GAN: Unsupervised Joint Face Rotation and De-Occlusion from a Single-View Image

[7]. StyleSDF: High-Resolution 3D-Consistent Image and Geometry Generation

[8]. An analysis of generative adversarial networks

and variants for image synthesis on the MNIST dataset

[9]. A Face-to-Face Neural Conversation Model

[10]. Face Space Representations in Deep Convolutional Neural Networks

[11]. Few-shot Knowledge Transfer for Fine-grained Cartoon Face Generation

[12]. Synthesizing Coupled 3D Face Modalities by Trunk-Branch Generative Adversarial Networks

[13]. Rebooting ACGAN: Auxiliary Classifier GANs with Stable Training

[14]. Generative Adversarial Networks (GANs): Challenges, Solutions, and Future Directions