Attention-based Image-to-Video Translation for Synthesizing Facial Expression using Generative Adversarial Network

Kidist Alemayehu Bedada

A Thesis Submitted to the Department of Computer Science and Engineering

School of Electrical Engineering and Computing

Presented in Partial Fulfillment of the Requirement for the Degree of Master's in Computer Science and Engineering

Office of Graduate Studies

Adama Science and Technology University

February, 2022

Adama, Ethiopia

Attention-based Image-to-Video Translation for Synthesizing Facial Expression using Generative Adversarial Network

Kidist Alemayehu Bedada

Advisor: Worku Jifara Sori (Ph.D)

A Thesis Submitted to the Department of Computer Science and Engineering
School of Electrical Engineering and Computing
Presented in Partial Fulfillment of the Requirement for the Degree of Master's in Computer Science and Engineering

Office of Graduate Studies
Adama Science and Technology University

February, 2022
Adama, Ethiopia

DECLARATION

I hereby declare that this Master Thesis entitled "**Attention-based Image-to-Video Translation for Synthesizing Facial Expression using Generative Adversarial Network "** is my original work. That is, it has not been submitted for the award of any academic degree, diploma, or certificate in any other university. All sources of materials that are used for this thesis have been duly acknowledged through citation.

Kidist Alemayehu _____ _____

Name Signature Date

RECOMMENDATION

I, the advisor of this thesis, hereby certify that I have read the revised version of the thesis entitled "**Attention-based Image-to-Video translation for synthesizing facial expression using Generative Adversarial Network**" prepared under my guidance by Kidist Alemayehu submitted in partial fulfillment of the requirements for the degree of Mater's of Science in Computer Science and Engineering. Therefore, I recommend the submission of a revised version of the thesis to the department following the applicable procedures.

Dr. Worku Jifara (Ph.D) _____ _____

Major Advisor Signature Date

APPROVAL SHEET

I, the advisor of the thesis entitled "**Attention-based Image-to-Video Translation for Synthesizing Facial Expression using Generative Adversarial Network**" and developed by **Kidist Alemayehu**, hereby certify that the recommendation and suggestions made by the board of examiners are appropriately incorporated into the final version of the thesis.

Dr. Worku Jifara (Ph.D) _____ _____

Major Supervisor Signature Date

We, the undersigned, members of the Board of Examiners of the thesis by **Kidist Alemayehu** have read and evaluated the thesis entitled "**Attention-based Image-to-Video Translation for Synthesizing Facial Expression using Generative Adversarial Network**" and examined the candidate during the open defense. This is, therefore, to certify that the thesis is accepted for partial fulfillment of the requirement of the degree of Master of Science in Computer Science Engineering

_____ _____ _____

Chairperson Signature Date

_____ _____ _____

Internal Examiner Signature Date

_____ _____ _____

External Examiner Signature Date

Finally, approval and acceptance of the thesis are contingent upon submission of its final copy to the Office of Postgraduate Studies (OPGS) through the Department Graduate Council (DGC) and School Graduate Committee (SGC).

_____ _____ _____

Department Head Signature Date

_____ _____ _____

School Dean Signature Date

_____ _____ _____

Office of Postgraduate Studies, Dean Signature Date

ACKNOWLEDGMENT

In the first place, I would offer my genuine gratitude to Almighty God, who has boundless gifts and opportunities for me. Besides, I might want to thank the Virgin Mary the Mother of God.

I would like to acknowledge and give my deepest gratitude to my advisor Dr. Worku Jifara (Ph.D.) for his guidance, brilliant comments, and support throughout my work.

Then, I would like to thank Prof. Yunkoo Chung (Ph.D.) for his useful comments and suggestions to better this study.

The deepest gratitude for Anteneh Tilaye (M.Sc.) for his brilliant comments, recommendations, and support throughout my work.

I would like to express my deepest gratitude to the Multimedia Understand Group for granting me access to the MUG facial expression dataset.

I would like to express my sincere gratitude to all individuals who were volunteered to be part of the Local dataset.

I would like to thank Khalid Mohammed for helping me with the preparation of the Local dataset.

Finally, I would like to thank all computer vision SIG members, my family, and friends for their support and motivation throughout my thesis work.

TABLE OF CONTENTS

ACKNOWLEDGMENT i

LIST OF TABLES vi

LIST OF FIGURES vii

LIST OF SAMPLE CODES ix

LIST OF ABBREVIATIONS AND ACRONYMS x

LIST OF NOTATIONS AND SYMBOLS xi

ABSTRACT xii

CHAPTER ONE 1

INTRODUCTION 1

1.1. Background of the Study 1

1.2. Motivation of the Study 3

1.3. Statement of the Problem 3

1.4. Research Questions 4

1.5. Objectives of the Study 4

1.5.1. General Objective 4

1.5.2. Specific Objectives 4

1.6. Scope and Limitation of the Study 5

1.6.1. Scope of the Study 5

1.6.2. Limitation of the Study 5

1.7. Beneficiaries of the Study 5

1.8. Organization of the Thesis 6

CHAPTER TWO 7

LITERATURE REVIEW 7

2.1. Introduction 7

2.2. Generative Adversarial Network 8
2.2.1. Conditional GAN 9
2.3. Image-to-image translation 10
2.4. Video Synthesis 11
2.4.1. Image-to-Video Translation 12
2.5. Facial Attribute Manipulation 13
2.6. Related Works 14
2.7. Summary of Related Work 17
CHAPTER THREE 20
RESEARCH METHODOLOGY 20
3.1. Chapter Overview 20
3.2. Datasets 20
3.2.1. MUG facial expression dataset 20
3.2.2. Local dataset 21
3.3. Data Augmentation 22
3.4. Pre-processing Techniques 23
3.4.1. Resize Image 23
3.4.2. Normalization 23
3.5. Development Tools 23
3.5.1. Hardware Tools 23
3.5.2. Software Tools 24
3.6. Baseline work 24
3.7. Evaluation Methods 25
3.7.1. Average Content Distance (ACD) 25
3.7.2. Structural Similarity Index Measure (SSIM) 26

3.7.3. Peak Signal to Noise Ratio (PSNR) 27
CHAPTER FOUR 28
PROPOSED ARCHITECTURE 28
4.1. Chapter Overview 28
4.2. Model Architecture 28
4.3. Generator Architecture 30
4.3.1. Self-Attention Mechanism 33
4.3.2. Channel-Attention Mechanism 34
4.4. Discriminator Network 35
4.5. Model Learning Functions 36
4.7. Training Pseudocode 37
CHAPTER FIVE 39
IMPLEMENTATION OF THE PROPOSED MODEL 39
5.1. Chapter Overview 39
5.2. Working Environment 39
5.3. Environmental Setup 39
5.4. Data Augmentation Implementation 40
5.5. Proposed Model Implementation 40
5.5.1. Self-attention Implementation 43
5.5.2. Channel Attention Implementation 43
5.5.3. Discriminator Network Implementation 44
5.6. Hyperparameter Configuration 45
5.7. Experiment Class 46
CHAPTER SIX 48
RESULTS AND DISCUSSIONS 48

6.1. Chapter Overview 48
6.2. Results of the Study 48
6.2.1. Data Augmentation results 48
6.2.2. Experimental Results 49
6.3. Sample training log for AI2VFEGAN 59
6.4. Evaluation Metrics 61
6.4.1. Average Content Distance (ACD) 61
6.4.2. Structural Similarity Index and Peak Signal to Noise Ratio 61
6.5. Results Discussion 62
6.5.1. Discussion on ACD scores result 62
6.5.2. Discussion on SSIM scores result 65
6.5.3. Discussion on PSNR scores result 66
CONCLUSION AND FUTURE WORKS 67
Conclusion 67
Future Work 68
REFERENCES 71
Appendix A: MUG facial expression database license agreement 79
Appendix B: Ablation Study 80
Appendix C: Result on different Epochs 82
Appendix D: Sample code 83

LIST OF TABLES

Table 2. 1 Summary of related works.....17
Table 3. 1 Dataset number of frames before and after augmentation……22
Table 3. 2 Hardware tools 23
Table 5. 1 Content of the generator architecture……...….41
Table 5. 2 Contents of the discriminator architecture 42
Table 5. 3 Hyperparameter configuration.....46
Table 5. 4 List of experiment class.....47
Table 6. 1 ACD-I, ACD-C, and ACD-G scores of the five experiments……..……61
Table 6. 2 SSIM and PSNR score of the five experiments.....62

LIST OF FIGURES

Figure 2. 1 GAN architecture 8
Figure 2. 2 CGAN architecture 10
Figure 2. 3 Image-to-Image translation 10
Figure 2. 4 Image-to-video translation using a driving video 13
Figure 3. 1 The proposed model process flow 20
Figure 3. 3 Local dataset sample 22
Figure 4. 1 Overall architecture of the proposed model 29
Figure 4. 2 Generator Architecture 32
Figure 4. 3 Self-attention GAN 33
Figure 4. 4 Channel attention 34
Figure 4. 5 Discriminator architecture 36
Figure 6. 2 Sample augmentation result on Local dataset 49
Figure 6. 3 Result of AffineGAN 50
Figure 6. 5 Generated frames with AffineGAN network 50
Figure 6. 6 Result of MAEGAN 51
Figure 6. 7 Generated frame with MAEGAN 52
Figure 6. 8 Result of MAEGAN+SA 52
Figure 6. 9 Generated frames with MAEGAN+SA 53
Figure 6. 10 Result of MAEGAN+SA+CA 53
Figure 6. 11 Generated frames with MAEGAN+SA+CA 54
Figure 6. 12 Result of AI2VFEGAN 54
Figure 6. 13 Generated frames with AI2VFEGAN 55
Figure 6. 15 Generated frames of four catagories on the Local dataset 56
Figure 6. 16 Result of anger expression on the Local dataset 57
Figure 6. 17 Generated image sequences with five frames 58
Figure 6. 18 Generated image sequences with three frames 59
Figure 6. 19 Generator loss graph 60
Figure 6. 20 Discriminator loss graph 60
Figure 6. 21 ACD-I scores of the five experiments 63

Figure 6. 22 ACD-C scores of the five experiments 64
Figure 6. 23 ACD-G scores of the five experiments 65
Figure 6. 24 SSIM score of the five models 65
Figure 6. 25 PSNR score of the five models 66

LIST OF SAMPLE CODES

Sample code 5. 1 Data augmentation implementation 40
Sample code 5. 2 Generator function implementation 41
Sample code 5. 3 Expression intensity value implementation 41
Sample code 5. 4 Self-attention implementation 43
Sample code 5. 5 Channel attention implementation 44
Sample code 5. 6 Discriminator network implementation 45

LIST OF ABBREVIATIONS AND ACRONYMS

2D	Two dimensional
3D	Three dimensional
ACD	Average Content Distance
AffineGAN	Facial Image-to-Video Translation by a Hidden Affine Transformation
ANN	Artificial Neural Network
AU	Action Unit
BCE	Binary Cross-Entropy
CGAN	Conditional Generative Adversarial Network
CK+	Extended Cohn-Kanade dataset
CNN	Convolutional Neural Network
CycleGAN	Unpaired image-to-image Translation Using Cycle-Consistent
GAN	Generative Adversarial Network
GIF	Graphics Interchange Format
GPU	Graphical Processing Unit
IBM	International Bussiness Machine
LSTM	Long Short Term Memory
MAE	Mean Absolute Error
MSE	Mean Square Error
MUG	Multimedia Understanding Group
RNN	Recurrent Neural Network
SAGAN	Self-Attention Generative Adversarial Network

LIST OF NOTATIONS AND SYMBOLS

Notation	Meaning
Dec	Decoder
D	Discriminator
e	Expression Intensity Value
Enc_b	Basic Encoder
Enc_r	Residual Encoder
fb	Basic Feature
fr	Residual Feature
ft	Target Feature
G	Generator
Io	Input neutral face image
It	Ground Truth Image
$\widetilde{It}$	Generated expression image
CA	Channel attention
SA	Self-attention
Z	Latent space
AI2VFEGAN	Attention-base Image-to-Video translation for Synthesizing Facial Expression using Generative Adversarial Network

ABSTRACT

Image-to-video translation seeks to generate a video clip from a single static image, unlike video prediction, its input is a static image with no temporal clues. The fundamental challenge in video generation is not only generating high-quality image sequences but also generating consistent frames with no abrupt shifts. With the development of Generative Adversarial Networks (GANs), great progress has been made in image generation tasks one of which is facial expression synthesis. This study focuses on generating facial expressions video from a single static image. Most previous works focused on synthesizing frontal and near frontal faces, which is not enough for many real-world applications. Some previous works manually assign expression intensity values to produce consistent image sequences, however, manual annotation fails when the video is incomplete. AffineGAN, a recent study, uses affine transformation in latent space to automatically infer expression intensity value, however, this work requires extraction of the feature of the target ground truth image, and there is also a quality issue on the generated image sequences. To address these issues, a new model is proposed by integrating self and channel attention mechanisms into the generator network for quality image generation, and Mean Absolute Error is utilized to train the network. This study also proposed to infer the expression intensity value automatically without the need to extract the feature of the ground truth images. The local dataset is prepared with frontal and with two different face positions on the left and right sides. Average Content Distance metrics of the proposed solution along with different experiments have been measured and the proposed solution has shown improvements. Based on SSIM and PSNR the proposed model improves the baseline work AffineGAN from 0.808 to 0.842 and from 32.922 to 33.574 respectively. This work concludes that using Mean Absolute Error loss to train the network and integrating self and channel attention into the generator network improves the quality of the generated images sequences and evenly distributing values based on frame size to assign expression intensity value improves the consistency of image sequences being generated and also enables the generator to generate different frame size videos while remaining within the range [0,1].

Keywords: Generative Adversarial Network, Image-to-video translation, AffineGAN, self-attention, channel attention

CHAPTER ONE

INTRODUCTION

1.1. Background of the Study

Computer Vision seeks to enable and configure machines to perceive the world in the same manner that humans do, and apply that knowledge to a variety of tasks and processes (such as Image Analysis, Image Recognition, and Classification). After the advancement of deep generative networks in computer vision, photo-realistic image generation (Goodfellow et al., 2020), which focus on generating realistic-looking images from random vector, and image-to-image translation (Isola et al., 2017; J. Zhu et al., n.d.) which is concerned with translating images from one domain to another, are two well-known examples that have been achieved. This study focuses on image-to-video translation specifically on generating image sequences of facial expression from a single neutral image.

Humans can imagine various scenes of what would be the next move based on a given still image, therefore allowing machines to learn such activity would be advantageous in a range of application sectors such as cinematography (Zhou et al., 2018), movie production, photograph technology, and e-commerce. Image-to-video translation seeks to generate video sequences from the single static image; unlike image-to-image translation, it adds another temporal dimension to deep models; and, unlike video prediction, its input is a static image with no temporal clues (Fan et al., 2019).

Facial expression is defined as the movement of facial muscles on the skin that can convey an individual's emotional state. Facial expression synthesis focuses on creating new face shapes from the given input face without changing the particular characteristics of the given face; if implemented as an add-on, it can improve facial recognition accuracy and gender classification (Agianpuye & Minoi, 2013). Facial expression synthesis has several applications, including robot and avatars animation**,** video game development, and Graphics Interchange Format (GIF) generation.

Earlier methods synthesize facial expression using traditional approaches including 2D/3D image warping (Blanz et al., 2003; Garrido et al., 2014), image reordering (K. Li et al., 2012; Yang et al., n.d.), or flow mapping methods (Yang et al., 2011; Yeh et al., 2016), the majority of which are example-based or morph-based. Recent methods rely on generative models to deal with facial expression synthesis. One of the most notable is GAN (generative adversarial network) (Goodfellow et al., 2020), which has made significant advances in image generation. (Bozorgtabar et al., 2020; M. Chen et al., 2018; Choi et al., 2018; Fu et al., 2021; Qiao et al., 2018; X. Wang et al., 2018; Yan et al., 2020) have addressed the issue of facial expression but limited to static facial expression generation.

Producing a facial expression video from one image is a one-to-many mapping problem in which the output has many more unknowns to solve than the input, which lacks temporal information. Recently, researchers addressed the challenge of image-to-video translation for generating facial emotions from a single picture, including facial geometry-based (Song et al., 2018), face parsing-based (Lu et al., 2018), action unit-based (Pumarola et al., 2018), and 3D blendshape model-based (Ververas & Zafeiriou, 2020). (Ding et al., 2018; Lu et al., 2018; Song et al., 2018) produce facial expressions images without background face (ear, hair, and half of the forehead). Other work (Fan et al., 2019) proposed a user-controllable framework to synthesize facial expressions and this approach assigns expression intensity value manually to make the generated image sequences consistent, however, the manual annotation fails when the video is incomplete.

To solve the manual annotation problem AffineGAN (G. Shen et al., 2019) employed an affine transformation to give an expression intensity to each facial image in the latent space, assuming that the ground truth (real) image features are equal to the produced image features; if they are not equal, inconsistency occurs. This effort also necessitates extracting features from the target ground truth images to compute the expression intensity value.

In addition to the above-stated problems, the majority of existing works only focus on synthesizing frontal and near-frontal face expressions. But the non-frontal face images also need to be synthesized to apply the model for most real-world applications. Some works (Fu et al., 2021; Qian et al., 2019) synthesize face expression and pose simultaneously but they disregard expression intensity variations.

This work proposes to generate good-quality and consistent image sequences of facial expression and varying frame size videos inspired by the AffineGAN (G. Shen et al., 2019) network. This study proposes to infer expression intensity value by evenly distributing values in the range [0,1] according to the number of frames in a given video to produce consistent image sequences and varying frame size videos, as well as adding attention mechanisms and constraints to the network architecture for better results.

1.2. Motivation of the Study

Recently, with the achievement of GANs (Generative Adversarial Networks), substantial improvements have been achieved in image generation tasks, one of which is facial expression synthesis. However, most of the proposed solutions have addressed the issue of static facial expression generation, while the problem of facial expression video generation is still less investigated. The ability to create a video sequence of face emotions from a single static image is valuable in a range of sectors, including anime film production and video game development. If implemented as an add-on, it can also increase the accuracy of facial recognition and the gender classification system.

Recent methods which tackled the image-to-video translation to synthesize facial expressions video were limited to frontal and near frontal face image synthesis, generating non-frontal facial expressions would make the video more realistic and applicable for various real-world applications. Another main issue in video generation is the consistency of image sequences, some methods (Fan et al., 2019; Lu et al., 2018) assign expression intensity values manually to the range [0,1] to make the generated image sequence consistent in time. However, the manual annotation failed when the video is incomplete. Recent work (G. Shen et al., 2019) assigns the expression intensity value automatically by applying affine transformation in the latent space. However, to determine the expression intensity value, this work must extract features of the target ground truth images. The above-stated concepts motivate me to work in this area.

1.3. Statement of the Problem

Generating a facial expression video from a single static image is a one-to-many mapping problem in which the output has a lot more unknowns to solve than the input which is missing temporal information. The main goal is not only to generate realistic image sequences but also consistent frames that vary slowly over time with no sudden transitions. In addition, the model

has to be able to create a video with a varied frame size which means the generated image sequence should be from neutral to peak expression even if the video frame size differs.

Most previous works that addressed the issues of generating video sequences from static face images were only focused on frontal and near frontal face position to synthesize facial expression. The non-frontal face image must also be synthesized to be used for diverse real-world applications.

To create consistent video frames, some works (Fan et al., 2019; Lu et al., 2018) give expression intensity values manually by their temporal dimension. However, manual annotation fails when the video is incomplete and temporal position could also be a poor proxy for expression intensity since it presupposes that the rate of expression change is constant over time. Recent work (G. Shen et al., 2019) utilized an affine transformation to give an expression intensity to each facial image in the latent space, assuming that the ground truth (real) image features are equal to the produced image features; if they are not equal, inconsistency occurs. This effort also necessitates extracting features from the target ground truth images to compute the expression intensity value and there is a quality problem on the generated image sequences. To overcome those problems Attention-based Image-to-video translation for synthesizing facial expression using GAN is proposed.

1.4. Research Questions

RQ1. How can expression intensity values be inferred to generate consistent image sequences?

RQ2. What are the effective constraints to improve the quality of the image sequences?

1.5. Objectives of the Study

1.5.1. General Objective

The general objective of the study is to generate consistent facial expression video from a single static image using the attention-based generative adversarial network.

1.5.2. Specific Objectives

The following are specific objectives addressed to achieve the general objective.

- To review concepts and related works on image-to-video translation for synthesizing facial expression

- To prepare a local dataset of four expressions with frontal and two different sides on both left and right side
- To assign expression intensity value by evenly distributing values according to the frame size in a given video
- To design GAN based model by adding learning constraints and attention mechanisms to generate quality image sequences
- To evaluate the performance of the baseline work and the designed GAN based model with ACD (Average Content Distance), SSIM (Structural Similarity Index Measure), and PSNR (Peak Signal to Noise Ratio) metrics

1.6. Scope and Limitation of the Study

1.6.1. Scope of the Study

The objective of this study is to create realistic-looking and consistent facial expression image sequences from a static neutral image and also to generate videos with varying frame sizes. The proposed model is conditioned with a neutral face image and expression intensity value. As a result, the scope starts from preparing a local dataset and facial expression image sequences generation to results evaluation in contrast to the previous approach.

1.6.2. Limitation of the Study

The limitation of the study includes:

- It does not involve the synthesis of facial expressions from one peak expression to another.
- It has only been done for the four expressions (anger, disgust, happiness, and surprise).

1.7. Beneficiaries of the Study

The proposed thesis work will have numerous benefits in the areas listed below.

- Facial recognition system: Facial expression synthesis can boost the performance of facial recognition systems; if implemented as an add-on, because even little expression alterations can considerably affect recognition accuracy.
- Anime production studio and Video game development: Cartoon and anime production companies can employ facial expression synthesis technologies to minimize the number

of frames needed to animate a facial emotion onto a character's face. Realistic facial emotions on 3D objects and avatars can be generated by video game developers.

- Human-Computer Interaction: Human-Computer Interaction can also benefit from facial expression synthesis. For example, front camera sensors on mobile devices could be used to detect the user's facial expression and infer his or her emotional state.
- GIF generation: GIFs are a sort of art that is widely used in internet communications and social media.

1.8. Organization of the Thesis

The remaining of this thesis is structured as follows:

Chapter Two: Discuss the background literature and related works regarding image-to-image translation, video synthesis, Image-to-video translation, facial attribute manipulation, and a summary of related works.

Chapter Three: Discuss research methodologies such as dataset collection, data preprocessing, development tools, baseline work, and evaluation metrics.

Chapter Four: Discuss the proposed model architecture, generator network, discriminator network, learning parameter, and the training pseudocode.

Chapter Five: Discuss the proposed model implementation, the working environment setup, the code snippet of the proposed solution, the hyperparameters used, and the experiment class.

Chapter Six: Explain the proposed solution findings, compare the new result with the prior solution, and explain the meaning of the findings.

Chapter Seven: This chapter concludes the research and provides recommendations for further work.

CHAPTER TWO

LITERATURE REVIEW

2.1. Introduction

Machine Learning is an Artificial Intelligence area that focuses on teaching machines to learn on their own without much human assistance. Arthur Samuel designed the first computer learning application, a game of checkers. The application improved IBM's computer as it played more. Following that, several machine learning technologies came in and out of fashion. One of the ML(Machine Learning) technology is Computer Vision, which allows machines to observe, understand, and interpret what they see (Gollapudi, 2019). Machine Learning techniques necessitate the assistance of a domain expert to determine the bulk of the applicable features to reduce data complexity and make patterns obvious to learning algorithms.

Deep learning, a sub-type of machine learning, focuses on self-improvement by analyzing computer algorithms. Since deep learning seeks to progressively learn high-level features gradually from data, the need for domain knowledge and intensive feature extraction is reduced. Deep learning makes use of artificial neural networks (ANN), which are meant to mimic how people think and learn. DL(Deep Learning) advancements in computer vision have been a huge success, particularly with CNN (Convolutional Neural Network) method. Yann LeCun created the first CNN, LeNet (Lecun et al., 1998), to perform character recognition tasks. Thanks to the availability of enormous datasets, such as the ImageNet, AlexNet (Krizhevsky Alex et al., 2012) was able to build sophisticated CNN that performed previously unachievable computer vision tasks.

Generative models are the most remarkable new deep learning methodologies. A generative model's purpose is to learn the real dispersion of the training set's data so that new data with some variability can be created. Generative Adversarial Network (Goodfellow et al., 2020) is a known deep generative model based on a game theory strategy that seeks to find the Nash equilibrium between two networks, Generator G and Discriminator D.

2.2. Generative Adversarial Network

Goodfellow proposed GAN (Goodfellow et al., 2020) in 2014. Generator G and Discriminator D are two interacting neural networks in GAN that are trained collaboratively by performing a zero-sum game in which the generator aims to produce fake data that looks like genuine data while the discriminator aims to distinguish between actual and produced data.

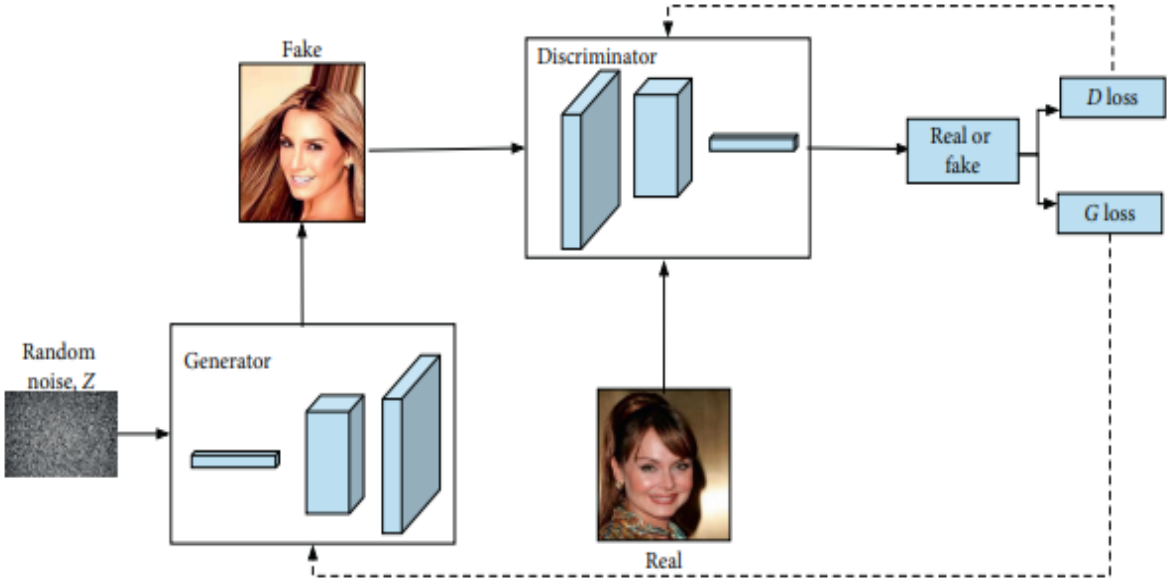


Figure 2. 1 GAN architecture

(Source: adapted from (Nandhini Abirami et al., 2021))

A discriminator network is a normal classifier model that attempts to distinguish actual data from data produced by the generator. During training, it uses real data distribution as positive examples and produced data distribution as negative examples. Two loss functions are linked with the discriminator: generator loss and discriminator loss. During the discriminator network training phase, the discriminator only utilizes the discriminator loss by overlooking the generator loss. If the discriminator incorrectly classifies created data as genuine, it will be penalized by the discriminator loss and vice versa. The discriminator then adjusts its weight through backpropagation to better distinguish genuine samples from produced samples. Generator network trains to generate fake samples that mimic real data from a given random vector. Its main objective is to trick the discriminator network to classify its outcome as real. The generator loss punishes the generator if it fails to trick the discriminator.

Zero-sum implies that, if the discriminator effectively distinguishes real and produced examples, it is remunerated, or no need to adjust the model parameters, though the generator is punished with significant changes to model parameters. And, when the generator tricks the discriminator, it is remunerated, or no need to adjust the model parameters, yet the discriminator is punished with significant changes to model parameters. At a cutoff, the generator produces ideal copies from the input domain without fail, and the discriminator is unable to distinguish and predicts "uncertain" (for example 50 percent real and fake) for each case. The core idea in GAN training is to minimize the discrepancy between genuine data distribution and fake data distribution.

The loss function of the GAN training:

$$Ex\left[\log\left(D(x)\right)\right] + Ez\left[\log\left(1 - D\left(G(z)\right)\right)\right] \qquad eq.(2.1)$$

There are different Generative Adversarial Network architectures Conditional GAN is one of them.

2.2.1. Conditional GAN

The generative models can be learned to produce new samples from the input domain, where the input, z (random vector), is supplemented with some extra information (Mirza & Osindero, 2014). Conditional GAN is a kind of GAN that is conditioned with additional information Y for both generator and discriminator (Y can be a class label, like male and female in the generating photographs of peoples or a digit in the generating of handwritten digits images).

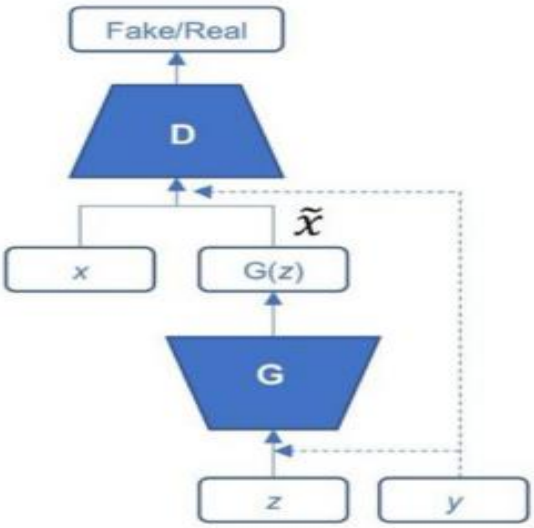


Figure 2. 2 CGAN architecture

(Source: adapted from (Z. S. Chen et al., 2021))

The GAN models can also be conditioned with input content, this permits applications of GAN like image-to-image translation, video generation, and image-to-video translations.

2.3. Image-to-image translation

A method for translating images from a given image domain into another, such as from label maps to pictures, sketches to shoes, summer to winter, is called image-to-image translation.

Pix2pix (Isola et al., 2017) proposed a model, which leverages CGAN for image-to-image translation. The pix2pix model uses a patch-wise discriminator (patchGAN), that attempts to discriminate every nearby image patch instead of the entire image. It significantly reduces the discriminator burden because discriminating local patches requires significantly less model capacity than discriminating entire images. Pix2pixHD (T. C. Wang et al., 2018) enhances pix2pix by including a multi-scale discriminator and coarse-to-fine generator, resulting in higher fidelity image translation.

Figure 2. 3 Image-to-Image translation

(Source: adapted from (Isola et al., 2017))

CycleGAN (J. Zhu et al., n.d.) proposed a method for translating images from a given domain A to target domain B without paired data. CycleGAN learns mappings in both directions at the same time and utilizes cycle consistency loss. (Isola et al., 2017; J. Y. Zhu, Park, et al., 2017) are limited to one-to-one translation but most real-world applications require one-to-many translation.

(J. Y. Zhu, Zhang, et al., 2017) intends to represent a dispersion of possible yields in a conditional generative modeling scenario. Given source input A it can generate diverse sets of output B's. (One & Translation, 2019; Yongqi Zhang, 2018) studied the issue of one-to-many unsupervised image translation, in which the input sample from one domain correlates to multiple samples from another.

2.4. Video Synthesis

There are three dimensions to image data (height, width, and channel), while video data has four dimensions (Height, Width, Channel, and temporal). Video is a collection of images that cannot be shuffled, the images should be consistent in the short term. Video synthesis is concerned with creating video content, rather than static images. Video synthesis, unlike image synthesis, must ensure that the output videos are consistent in time. Unconditional and conditional video synthesis are common classifications for video synthesize (Cheong, n.d.).

Unconditional video synthesis uses random noise to generate sequences (Saito et al., 2017; Tulyakov et al., 2018). (Saito et al., 2017) makes use of two distinct generators: an image generator and a temporal generator. The temporal generator accepts one latent vector as input and produces a group of latent vectors, every of which relates to image frames in a video. A collection of such latent vectors is transformed into a video by the image generator. MocoGAN (Tulyakov et al., 2018) decomposes the sequence's movement and content parts and generates the motion using a fixed latent space and a succession of latent spaces. Because such a method must model all the spatial and temporal content in an exceeding video, the resulting videos are frequently short or have very constrained motion patterns. Unconditional video generation is not yet mature enough to be used in a practical setting (Cheong, n.d.).

Conditional video synthesis, on the alternative hand, is conditioned on input content and thus produces higher-quality results. A common category is future frame prediction (Cai et al., 2018; Denton & Birodkar, 2017; Kalchbrenner et al., 2017; Lee et al., 2018; Liang et al., 2017; Lotter

et al., 2017; Mathieu et al., 2016; Reda et al., 2018; Walker et al., 2017) which attempts to forecast the subsequent frame sequences based on past frames. (Liang et al., 2017) uses the dual adversarial training approach to solve the primal future frame prediction and future flow prediction tasks at the same time. (Villegas, Yang, Hong, et al., 2017) proposed a motion-content model for forecasting future frame sequences at a pixel level. In (Villegas, Yang, Zou, et al., 2017) the authors use LSTM to predict future poses after observing numerous consecutive poses. Most previous prediction techniques (Cai et al., 2018; Reda et al., 2018; Villegas, Yang, Hong, et al., 2017; Villegas, Yang, Zou, et al., 2017) are temporal-based, which requires forecasted videos to be pre-determined amount of frames so once the models have been trained, users are unable to change the temporal lengths or variable speeds of the videos.

On the alternative hand, conditional video synthesis conditioned on input content video that shares the same high-level representation is another common form of conditional video synthesis. This body of work has yielded promising results on a range of tasks, such as converting high-level representations into photorealistic videos (T. Wang et al., n.d.) and animating characters with new motions (Chan et al., 2019).

2.4.1. Image-to-Video Translation

In contrast to video prediction, the image-to-video translation uses a single static image as input rather than video frames. (Siarohin, Lathuiliere, et al., 2019; Siarohin, Lathuilière, et al., 2019), decouple appearance and movement information to animate a single input according to the movement of the driving video. (L. Zhao et al., 2018) introduced a two-stage video production paradigm in which image sequences are created from structures and then refined using temporal signals. 3D Morphable Model was utilized for face expression retargeting, and LSTM was employed for human pose predicting.

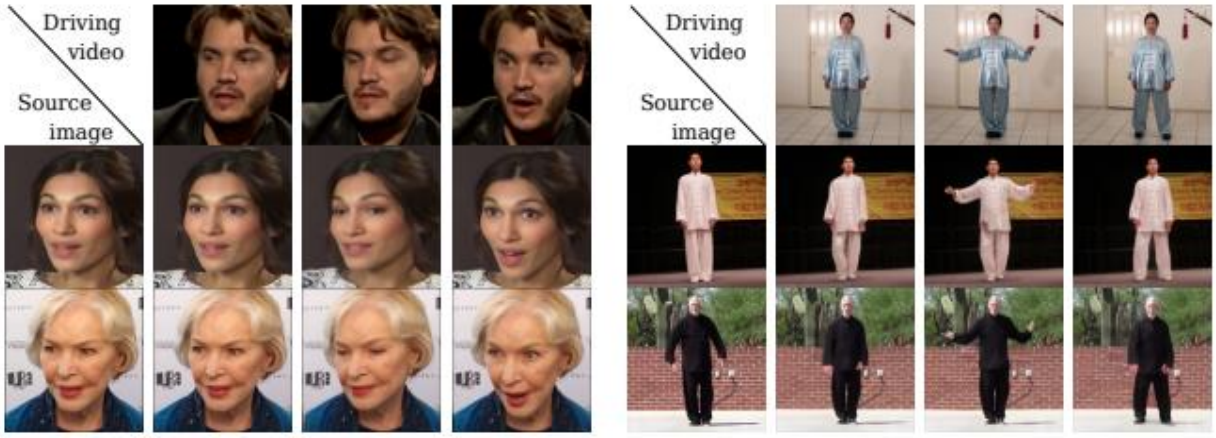


Figure 2. 4 Image-to-video translation using a driving video

(Source: adapted from (Siarohin, Lathuilière, et al., 2019))

(B. Chen et al., 2017) introduced a framework that generates different videos from a single image. It employs transformation generation for modeling motion between frames. (Zhou et al., 2018) produced cinematograph sequences from one image. To produce cinematography, they combine generative models with RNN. (Y. Li et al., 2018) divides the work of prediction future frames into two steps flow prediction and flow-grounded frame generation. The flow prediction phase models the spatial-temporal link as well as future uncertainty. The frame generation procedure assures that expected sequences have the same manifold structure as real sequences. Both (Y. Li et al., 2018; L. Zhao et al., 2018) are temporal-based video predictions. However, because only a single image is detected in image-to-video translation, temporal-based approaches struggle to obtain enough sequence information. Furthermore, because of the limitation of LSTM, temporal-based algorithms could only generate a limited amount of frames.

2.5. Facial Attribute Manipulation

Facial attributes are natural semantic characteristics of face images that depict human-understandable visual properties like smiles, mustache, and glasses. Facial Attribute Manipulation uses generative models to modify a face image to synthesize or detach desired attributes (Zheng et al., 2020).(Choi et al., 2020; M. Li et al., 2016; Perarnau et al., 2016; W. Shen & Liu, 2017) attempted to switch attribute classes such as aging, adding glasses, gender-swapping, and changing the hair color. (Bao et al., 2018) unravel facial identities and attributes for identity-preserving face generation by combining different identities and attributes. (Nirkin et al., 2019) perform face recreation and face-swapping.

Some methods synthesize facial expression using traditional approaches including 2D/3D image warping (Blanz et al., 2003; Garrido et al., 2014), image reordering (K. Li et al., 2012; Yang et al., n.d.), or flow mapping methods (Yang et al., 2011; Yeh et al., 2016), the majority of which are example-based or morph-based. (Choi et al., 2018) is an approach for performing image-to-image translations on facial expression synthesis across multiple domains with a single model. From a given input face image with a specific expression and a target expression from another person, (Qiao et al., 2018) produces a face with target expression while retaining the input face's identity. (Fu et al., 2021) perform image-to-image translation to generate facial expressions and pose simultaneously. (Qian et al., 2019) synthesis face with different expressions from boundary image. However, (Fu et al., 2021; Qian et al., 2019) disregard expression intensity variations.

2.6. Related Works

For animating still face images, (Averbuch-Elor et al., 2017) used a 2D warping technique to transfer the subject's expression from the reference video to the given image. The model begins by transferring geometric motion from the source face's landmark changes to the input neutral frontal face. After a succession of 2D affine warping, the appearance and head pose can be animated.

(Olszewski et al., 2017) use per-frame source appearance and a single input texture to train a deep generative network to derive realistic per frame deformation of the input identity. Then from the source expression geometry and newly derived texture, the face animated and video replacement performed on the given video using target appearance. Both (Averbuch-Elor et al., 2017; Olszewski et al., 2017) synthesize video sequences of single face images based on a reference video but the reference video could provide a lot of information.

(Y. Zhao, 2018) proposed the factored conditional restricted based Boltzmann machine and reconstruction contractive based auto-encoder (FCRBM-RCAE) framework for facial expression animation from single frontal face images and emotion labels. But this approach only generates facial expression without background face(forehead, ear, and hair) and the subject identity in a sequence often changes.

ExprGAN (Ding et al., 2018) generates facial expressions of varying intensities based on the source image and expression label. To learn expression code, an expression controller module was proposed, and a regularizer network is also used to increase the mutual information between

the expression code and produced image. However, there is a problem with maintaining identity information and local details such as hair.

G2-GAN (Song et al., 2018) employed facial geometry to guide photo-realistic facial expression synthesis. A pair of GANs are trained to perform opposing tasks: expression synthesis and expression removal, forming a mapping loop between expressionless and expressive faces. However, target fiducial points need to be provided for synthesizing designated expressions.

CAFP-GAN (Lu et al., 2018) synthesizes facial expressions using a face parsing map. This method includes two subnetworks: the face parsing prediction network (FPPN) and the facial expression synthesis network (FESN). The FPPN predicts the face parsing map based on the given neutral image, expression label, and intensity value. The FESN predicts the target picture based on the created face parsing map and the neutral image. However, this work arranged the intensity value manually. The manual annotation failed when the video is incomplete.

GANimation (Pumarola et al., 2018) uses Action Units(AU) that portray the anatomical face motions in a continuous manifold of motions. Cycle consistency loss was also utilized to maintain the identity of each individual. However, their pipeline entails the utilization of a detector to localize and crop out the faces in the image.

In (Fan et al., 2019) the authors introduced a user-controllable method for creating video clips from a profile face image. A training video's expression intensities are manually labeled by their temporal positions like $e = [0, 0.1, 0.2, \ldots 1]$. However, the manual assumption fails when the video is incomplete and it could also be a poor proxy for expression intensity since it presupposes that the rate of expression change is constant over time.

To solve the manual assumption problem (G. Shen et al., 2019) proposed to infer the expression intensity value automatically by applying an affine transformation to each image in the latent space. This is done by subtracting the original image latent code from the target image latent code to infer the intensity, assuming that the latent code of the image that would be generated is equal to the ground truth latent code. Inconsistency will occur if they are not equal. This work also required the extraction of features from the target ground truth image to calculate the expression intensity value and the quality of the produced frames reduces as the expression grows.

A novel framework for creating six facial expressions from one image was proposed by (OTBERDOUT et al., 2020). The authors proposed using face geometry to demonstrate the movement of facial geometry as curves represented as a point in the hypersphere. For movement generation, the model learns the dispersion of facial expression dynamics using conditional Wasserstein GAN. Another conditional GAN is used to convert the generated movement to landmark sequences and then to picture sequences by altering the texture information. This technique, however, is limited to frontal facial expression synthesis.

The SliderGAN (Ververas & Zafeiriou, 2020) method was proposed for modifying expression (i.e., expression transfer) in face images that is driven by a series of statistical blendshapes. DCNN, which is useful for expression and speech transfer, was employed in this approach to predict the belendshape vector of the target image. However, it has a limitation on generating extreme expressions and does not always preserve the identity of the given image.

2.7. Summary of Related Work

Table 2. 1 Summary of related works

Related papers	Dataset used	Methodology	Limitation
(Ding et al., 2018)	Oulu-CASIA NIR-VIS facial expression database	ExprGAN, Adversarial Autoencoder	o There is a problem with maintaining identity information and local details such as hair o Does not consider the background face (ear, hair, and half of the forehead) o Focus only on the frontal facial expression synthesis
(Song et al., 2018)	CK+ and Oulu-CASIA NIR-VIS facial expression database	G2-GAN, facial geometry as a controllable condition	o Target fiducial points need to be provided for synthesizing target expressions. o Does not consider the background of the image o Focus only on the frontal facial expression synthesis
(Lu et al., 2018)	CK+, Oulu-CASIA NIR-VIS facial expression database, and CAS-PEAL-R1 Database	CAFP-GAN, Face parsing map as a controllable condition	o Fails when the video is incomplete o Does not consider the background face (ear, hair, and half of the forehead) o Focus only on the frontal facial expression synthesis

(Pumarola et al., 2018)	Emotionet	CycleGAN, AU annotation as a controllable condition	○ Require external tool for localizing and crop face ○ Focus on the frontal and near frontal facial expression synthesis
(Fan et al., 2019)	CK+ and CK++	Conditional GAN	○ Fails when the video is incomplete ○ Assume rate of expression change constant over time ○ Focus on the frontal facial expression synthesis
(G. Shen et al., 2019)	CK-mixed and Cheeks&Eyes	AffineGAN	○ Require feature extraction of target ground image to calculate expression intensity value ○ Focus on the frontal facial expression synthesis
(Ververas & Zafeiriou, 2020)	Emotionet	SliderGAN, 3D blendshape model as a controllable condition	○ Does not always retain the identity of a given face

Some research (Fan et al., 2019; Lu et al., 2018) manually assign expression intensity levels based on their temporal dimension to generate consistent image sequences. However, manual annotation fails when the video is incomplete. To solve the manual annotation problem AffineGAN (G. Shen et al., 2019) employs affine transformation in the latent space to assign expression intensity levels automatically. However, it required the extraction of features from the target ground truth image to calculate the expression intensity value.

In addition, most prior efforts on producing video sequences from static face images only focused on frontal and near-frontal face position to synthesize facial emotion. This paper proposes attention-based image-to-video translation using GAN to solve the manual annotation problem and also the consistency and quality problems on the generated image sequences.

CHAPTER THREE

RESEARCH METHODOLOGY

3.1. Chapter Overview

This chapter explains how the research methodology is carried out, including the datasets used, augmentation techniques, preprocessing techniques, development tools, baseline work, and evaluation metrics.

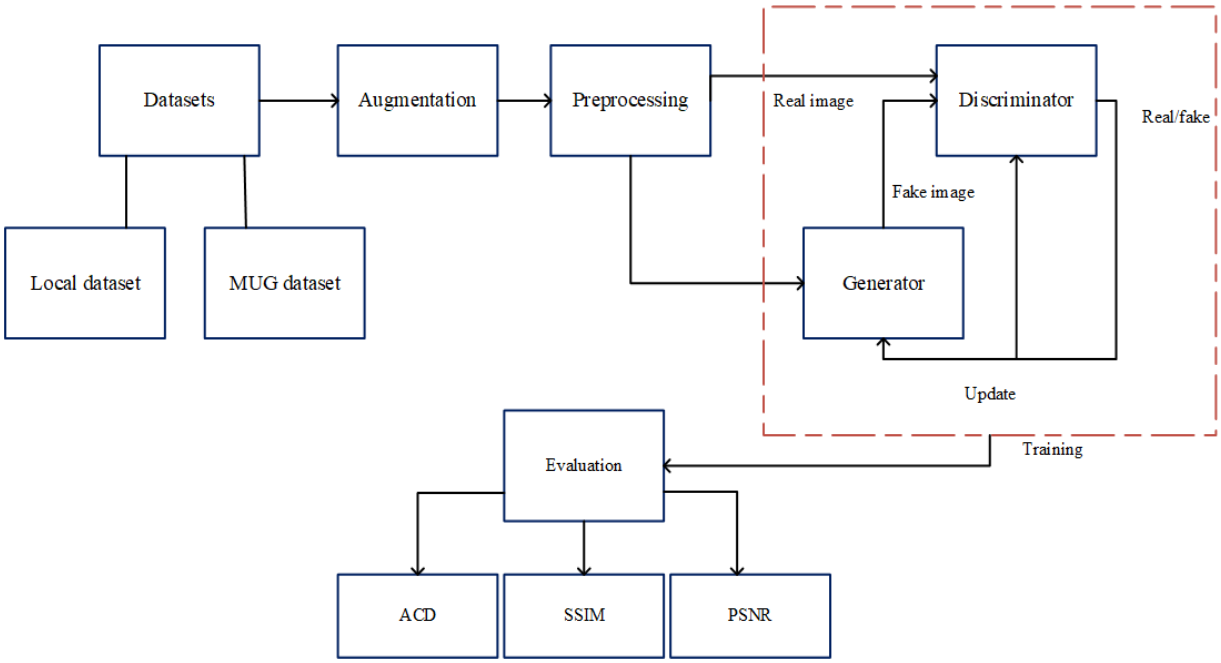


Figure 3. 1 The proposed model process flow

3.2. Datasets

Deep learning models rely heavily on data to learn; without data, the model cannot learn. The proposed model was trained and tested using the MUG (Multimedia Understanding Group) facial expression dataset and a local dataset that was collected by the researcher.

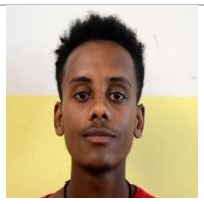
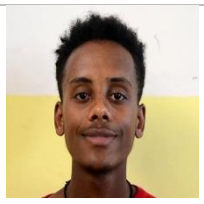
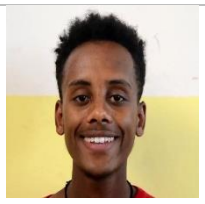
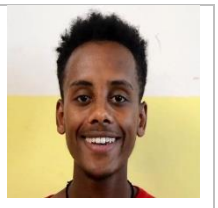
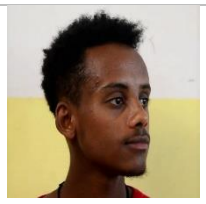
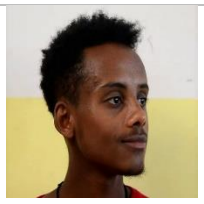
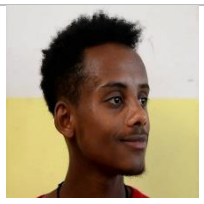
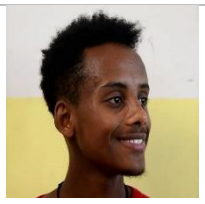
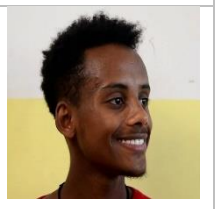
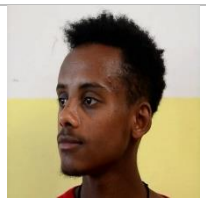
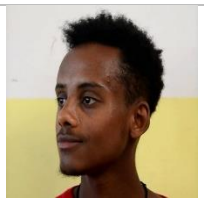
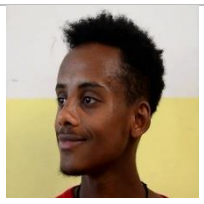
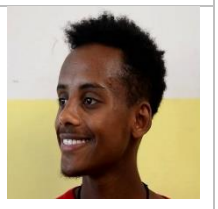
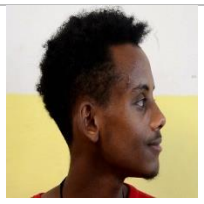
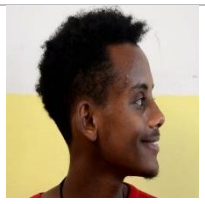
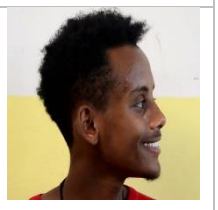
3.2.1. MUG facial expression dataset

It is a database developed by the Multimedia Understanding Group to aid in the investigation of facial expression recognition (Aifanti et al., 2010). It comprises image sequences that track the onset, peak, and offset temporal pattern of 86 participants in frontal position showing six

different facial expressions (disgust, anger, fear, sadness, happiness, and surprise). Images of 52 subjects, on the other hand, are only available to authorized users. This study made use of the expressions of anger, disgust, happiness, and surprise. Image sequences from the onset(neutral) to the peak of the temporal pattern were chosen as the dataset for each participant, and blurred frames were eliminated.

3.2.2. Local dataset

Since the MUG dataset only comprises image sequences of frontal facial expressions, a local dataset was collected. To construct the local dataset, 30 people were video recorded while displaying four different facial expressions (disgust, anger, surprise, and happiness) in frontal position and with two different face positions on both left and right sides. The video was cropped to make the faces nearly centered. The cropped video changed to image sequences. Image sequences from the onset(neutral) to the peak of the temporal pattern were chosen as the dataset for each participant, and blurred frames were eliminated.

(a)					
(b)					
(c)					
(d)					

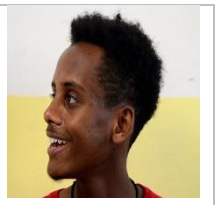
(e)					

Figure 3. 2 Local dataset sample

(a) Frontal position (b) Left side near 45-degree position (c) Right side near 45-degree position (d) Left near 90-degree position (e) Right near 90-degree position

The MUG dataset is appropriate for this study because it contains high-quality, well-organized video frames from neutral to peak expression. And the local dataset was created using the MUG dataset as a reference. The dataset is organized by expression, and the number of frames varies from person to person. The proposed model was tested using neutral images of 8 subjects from the MUG dataset, 4 subjects from the Local dataset, the rest subjects used for training the model. Table 3.1 shows the total number of frames used to train each expression.

3.3. Data Augmentation

Using image data augmenting, a training set can be made larger by generating different versions of the images. The fit models' ability to generalize what they have learned to new images is enhanced when they are trained on more data. Image augmentation techniques can generate a variety of image variants. The following image augmentation techniques were used: horizontal flip, random brightness, random saturation, adjusted brightness, and random contrast. Horizontal flip is a position augmentation that is used to flip the image horizontally and returns a result that is plausible in the real world. Random brightness and adjusted brightness techniques were used to change the image's brightness based on the brightness factor, brightness factor 32/255, and 0.3 utilized respectively. Random saturation and random contrast are used to randomly change the saturation and contrast of the image based on the saturation and contrast range respectively. Both the saturation and contrast ranges are set to the range of [0.5,1.5]. Other than horizontal flip, the other augmentation technique used in this thesis is color augmentation techniques because those are the effects that can happen naturally.

Table 3. 1 Dataset number of frames before and after augmentation

Expression	Total number of original frames		Total number of augmented frames		Total number of frames for each expression
	MUG	Local	MUG	Local	
Anger	654	1898	3270	9490	15312
Disgust	616	1725	3080	8625	14046
Happy	615	2214	3075	11070	16974
Surprise	540	1947	2700	9735	14922

3.4. Pre-processing Techniques

3.4.1. Resize Image

Because neural networks only take input of the same size, all photos in the datasets must be resized to a similar size. Since the original images in both datasets are of varying sizes, they must be resized to meet the network's input size. To reduce computational complexity, the images in the dataset were resized to $256x256$ following the work (G. Shen et al., 2019).

3.4.2. Normalization

Normalization aims to translate features to a common scale. This improves the model's training stability and performance. Normalization converts the natural range of feature values to the standard range $[-1,1]$. Image normalization is calculated as follows:

$$output = \frac{input_{image} - mean}{std} \qquad eq.(3.1)$$

Where $mean = 0.5$ and $std = 0.5$, $input_{image}$ in their natural range

Then, $output$ will be in the range $[-1,1]$

3.5. Development Tools

3.5.1. Hardware Tools

The hardware tools used for the research are listed in the table below.

Table 3. 2 Hardware tools

Number	Device Name	Usage for the research
1	Camera	To collect the local dataset
2	GPU	To train the model
3	Hardisk	To store the datasets

3.5.2. Software Tools

A group of computer programs used to create, manage, debug, or support other applications and programs are referred to as software tools.

Anaconda: is a dispersion of the R and Python programming languages that simplifies package management and deployment. Anaconda is extremely well-known since it simply needs to be installed once to provide a variety of tools used in ML and data research. Anaconda's package management, conda, enables the installation of libraries; it also includes a helpful interaction with pip, which allows for the installation of libraries that are not accessible through the anaconda package manager. It is also platform-independent, so it can be utilized on Windows, macOS, or Linux.

Jupyter notebook: An open-source, web application that integrates software code, explanatory text, computational output, and multimedia resources into a single document. It is integrated with the anaconda environment and used for editing code and seeing results.

PyTorch: is an open-source ML library that relies on a torch library that is utilized for applications, for example, computer vision and NLP. Because its syntax is similar to the Python programming language, Pytorch is nearly easier to learn.

Tensorflow: is an open-source toolkit that can be used for large-scale numerical computations as well as deep-learning applications. In this study, TensorFlow is used for data augmentation implementation.

3.6. Baseline work

To validate the performance of the proposed model, it compared with a recent method that translates a single static face image into facial expression sequences using GAN. As a result, the

AffineGAN architecture (G. Shen et al., 2019) was chosen as a baseline for this research because this work infers the intensity value automatically and generated good results compared to others.

The AffineGAN model constructs from one generator and three discriminators. The generator is constructed as a U-Net generator. And the discriminators are global discriminator, local discriminator, and alpha discriminator. The global discriminator contrasts the ground truth image with the generated image. The local discriminator is utilized to contrast the informative region of the generated image with the ground truth (real) image. The alpha discriminator was used to control the expression intensity value from deviating away from the range [0,1]. To automatically assign expression intensity value, AffineGAN proposed to subtract neutral image latent space from ground truth latent space by assuming the generated image latent space is equal to the ground truth latent space.

3.7. Evaluation Methods

An evaluation metrics measure the performance of a machine learning model. The evaluation of ML models is crucial for every project. The model was evaluated using ACD (Average Content Distance), SSIM (Structural Similarity Index), and PSNR (Peak Signal to Noise Ratio).

3.7.1. Average Content Distance (ACD)

Using OpenFace (Baltruˇ, n.d.), a 128-dimensional feature vector representing the content in that frame is generated for each frame. This is referred to as a content vector. The $l2$ distance between the content vectors of successive frames is then computed, and their average is taken. This resulted in the average content distance. If OpenFace cannot detect any faces from the anticipated frames, this indicates a poor prediction. To assess the quality of face identities, ACD-I computes the average distance between the original image and the output frames.

$$d_i = \sqrt{\sum_{j=0}^{n} (b_j - f)^2} \qquad eq.(3.2)$$

$$\text{ACD} - \text{I} = \frac{1}{n} \sum_{j} d_i \qquad eq.(3.3)$$

Where b_j is the feature of generated frames, f is the feature neutral face image, and n is the number of frames

To assess content consistency, ACD-C computes the average distance between all possible pairs of frames in a video.

$$d_c = \sqrt{\sum_{j=0}^{n} (b_j - b_{j+1})^2} \qquad eq.(3.4)$$

$$\text{ACD} - \text{C} = \frac{1}{n * (n-1) * 2} \sum_i d_c \qquad eq.(3.5)$$

To evaluate expression changes, ACD-G computes the average frame-to-frame distance between the generated frames and the corresponding ground-truth ones.

$$d_g = \sqrt{\sum_{j=0}^{n} (a_j - b_j)^2} \qquad eq.(3.6)$$

$$ACD - G = \frac{1}{n} \sum_j d_g \qquad eq.(3.7)$$

Where a_j is the feature of the real frames

3.7.2. Structural Similarity Index Measure (SSIM)

The SSIM is an approach for predicting images and videos quality. The SSIM is a complete reference metric, which means that image quality is assessed or predicted using an original uncompressed or distortion-free picture as a reference. It's a perceptual model that incorporates image deterioration as a perceived change in the structure of the information while also including crucial perceptual phenomena such as luminance and contrast masking components. The concept of structural information refers to the idea that pixels have substantial interdependencies, especially when they are spatially adjacent. These dependencies include crucial information on the structure of the visual scene's objects. Luminance is a phenomenon in which the visual distortions become less noticeable in bright regions, whereas contrast is a phenomenon in which the visual distortions become less noticeable in areas of strong activity in the image (Z. Wang et al., 2004).

$$l(x, y) = \frac{2\mu_x\mu_y + c1}{\mu_x^2 + \mu_y^2 + c1} \qquad eq.(3.8)$$

Where $c1 = (k1 * L)^2$ is nonnegative regularization constant used to avoid instability when the mean is close to zero, L is the dynamic range value, $k1$ is a small constant value

$$c(x, y) = \frac{2\sigma_x\sigma_y + c2}{\sigma_x^2 + \sigma_y^2 + c2} \qquad eq.(3.9)$$

Where $c2 = (k2 * L)^2$ is nonnegative regularization constant used to avoid instability when the standard deviation is close to zero, $k2$ small constant value

$$s(x, y) = \frac{\sigma_{xy} + c3}{\sigma_x + \sigma_y + c3} \qquad eq.(3.10)$$

Where μ_x, μ_y are local means, σ_x, σ_y are standard deviations, and σ_{xy} cross-covariance for images x, y, $c3 = c2/2$

$$SSIM(x, y) = [l(x, y)]^{\alpha}\,[c(x, y)]^{\beta}\,[s(x, y)]^{\gamma} \qquad eq.(3.11)$$

Where $\alpha, \beta,$ and γ are parameters that are used to adjust the relative importance of the three components.

3.7.3. Peak Signal to Noise Ratio (PSNR)

PSNR is the ratio of a signal's peak possible value to the possible value of noise that impacts the quality representation (ratio between the largest possible value and the smallest possible values). The PSNR is calculated using a logarithmic decibel scale.

$$PSNR = 20\log_{10}\frac{MAX_f}{\sqrt{MSE}} \qquad eq.(3.12)$$

Where MAX_f is the Maximum signal value and MSE is the Mean Square Error

$$MSE = \frac{1}{mn}\sum_{0}^{m-1}\sum_{0}^{n-1}\left||f(i,j) - g(i,j)|\right|^2 \qquad eq.(3.13)$$

Where f represents the original image, g represents the synthesized image, m is the number of rows and n is the number of columns of a pixel in the image respectively.

CHAPTER FOUR

PROPOSED ARCHITECTURE

4.1. Chapter Overview

This chapter describes the Attention-Based Image-to-Video Translation for Synthesizing Facial Expression using GAN(AI2VFEGAN) architecture with necessary diagrammatic representation and mathematical expression to image-to-video translation for synthesizing facial expression problems for improving the quality of the image sequences and consistency among them. This chapter is divided into four sections; the first section of this chapter describes the overall architecture of the proposed model to translate the image into video sequences. The second section discusses the generator and the discriminator architecture. The third section discusses the model learning functions and finally, the training pseudocode is explained.

4.2. Model Architecture

"Attention-based Image-to-Video translation for synthesizing facial expression using Generative Adversarial Network" is the proposed model that is utilized to address generating realistic-looking image sequences that are consistent in time. The proposed model has been adapted from "Facial Image-to-Video Translation by a Hidden Affine Transformation" for learning image-to-video translation. But instead of three discriminators like AffineGAN (G. Shen et al., 2019), the proposed model uses one discriminator network. Attention mechanisms were also introduced to the generator network and expression intensity value inferred by evenly distributing values according to the frame size in a given video.

The proposed model is built with one generator and one discriminator. The generator network was built as U-Net (Weng & Zhu, 2021) architecture. The discriminator network is a patchGAN discriminator following the work (Isola et al., 2017) that penalizes structure only at the scale of nearby image patches.

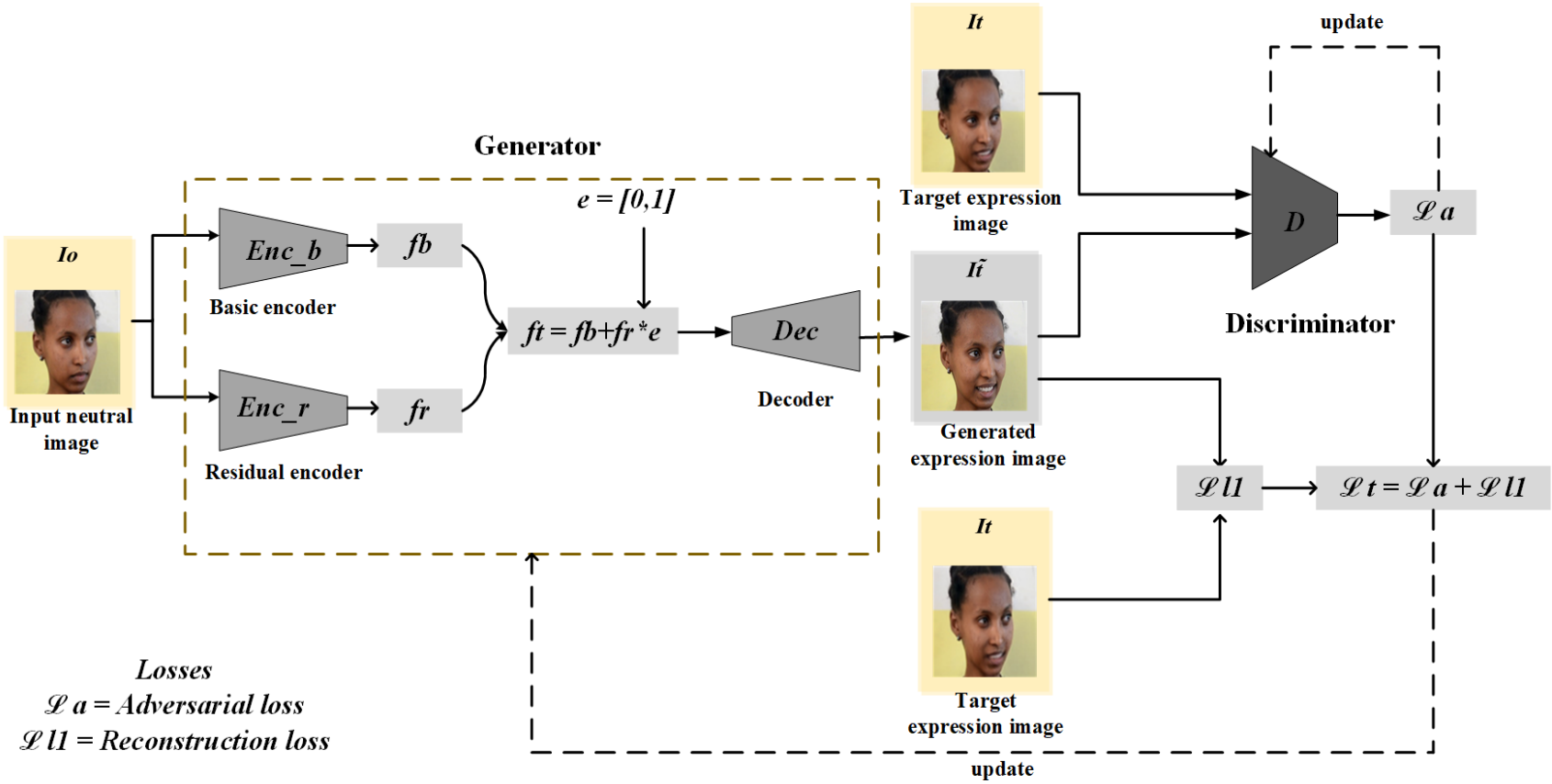


Figure 4. 1 Overall architecture of the proposed model

The generator, as illustrated in the figure. 4.1, is made up of two encoders, basic encoder Enc_b, and residual encoder Enc_r, as well as a decoder Dec. Both encoders are similar in structure and take the neutral image Io as input. The basic encoder is used to retain the feature of the neutral picture, while the residual encoder is utilized to capture the expression shift from neutral to peak expression. fb and fr are the features from the basic and residual encoder respectively.

The expression intensity e is used to control expression change and is calculated by equally distributing values within the range [0,1] based on the number of frames in the training dataset. In essence, you provide the beginning and endpoints of an interval, as well as the number of total breakpoints you want inside that interval. The interval is the number of frames in a specific video, and the starting and finishing points are set to 0 and 1, respectively. For example, if the frame size (interval) is five the intensity value will be $[0, 0.25, 0.5, 0.75, 1]$, expression intensity helps the network to learn that the expression should go from neutral to the peak.

Then target feature ft got from fb , fr and intensity value e.

$$Enc\_b(Io) = fb \qquad eq.(4.1)$$

$$Enc\_r(Io) = fr \qquad eq.(4.2)$$

$$ft = fb + fr * e \qquad eq.(4.3)$$

Finally, the target image $\widetilde{It}$ is generated by feeding target feature ft to the decoder Dec.

$$\widetilde{It} = Dec(ft) \qquad eq.(4.4)$$

when $e = 0$, $ft = fb$, the generator has to generate a neutral face image, and when $e = 1$ the generated face image should be at the peak expression state.

The discriminator network is utilized to contrast the produced expression image from that of the real expression image.

4.3. Generator Architecture

The generator network is utilized to create realistic-looking facial expressions with varying intensities. Both the encoders and the decoder are constructed as seven-layer neural networks with skip connections based on U-Net architecture (Weng & Zhu, 2021). U-Net, whose architecture is shaped like the letter 'U,' was originally designed for medical image

segmentation. It produced such good results that it was later used in other applications. In U-Net architecture, there are two pathways. The first path is the contraction path (encoder), which is utilized to record the context of the image. The expanding path (decoder) is utilized to achieve exact localization via transposed convolutions. To extend a feature to a target picture, U-Net employs the same feature maps that were utilized for contraction. This would preserve the image's structural integrity while drastically reducing distortion.

Both the encoders contain Convolution, LeakyReLU activation function, and Instance normalization except the first and the last layer, it contains only the Convolution at the first layer and Convolution and LeakyRelu activation function at the last layer. The decoder contains Transposed convolution, ReLU activation function, and Instance normalization except for the last layer it only contains transposed convolution and ReLU activation function.

The self-attention layer is incorporated at the first layer of the decoder to aid the generator to create images with fine details in every location that are precisely coordinated with fine features in distant parts of the image. Except on the first and last layer of the decoder, channel attention was added to help the network to focus on crucial features while filtering out irrelevant features.

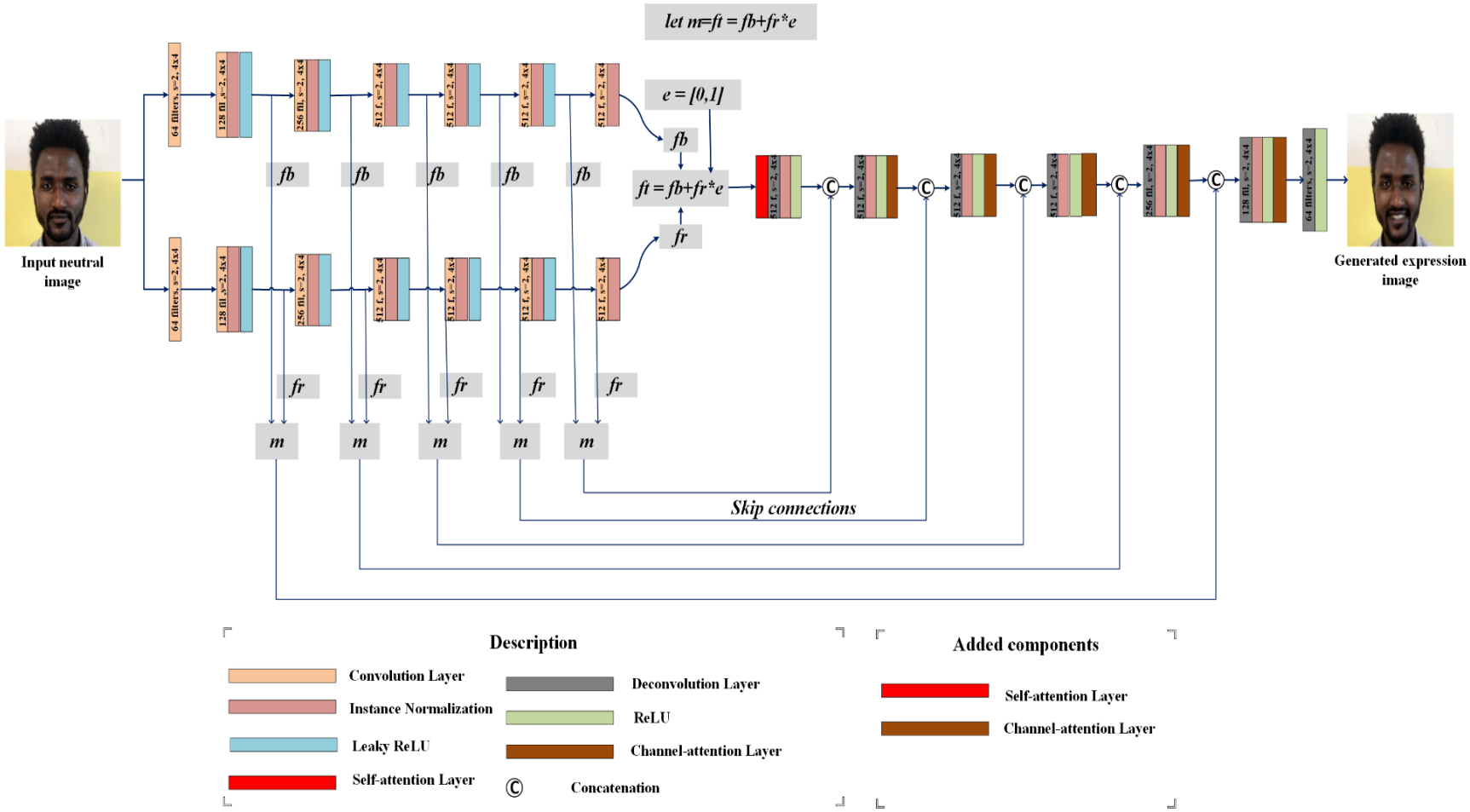


Figure 4. 2 Generator Architecture

4.3.1. Self-Attention Mechanism

The attention mechanism originated in NLP as an upgrade over the encoder-decoder-based neural machine translation system. Later, this approach, or variants of it, were applied in various applications such as computer vision.

Convolutional layers are utilized in the construction of the majority of GAN-based image generating models. Convolution analyzes information in a small neighborhood, hence representing long-range dependencies in pictures with convolutional layers only is computationally inefficient. SAGAN (H. Zhang et al., 2019) presents a self-attention method into convolutional GANs. With the self-attention method, the generator can produce images with fine details in every location that are precisely coordinated with fine features in distant parts of the image.

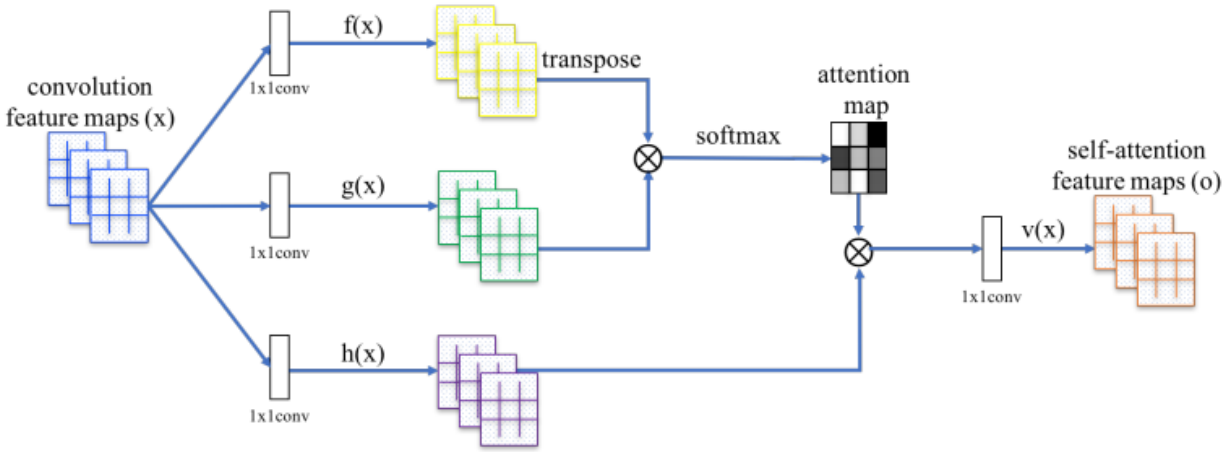


Figure 4. 3 Self-attention GAN

(Source: adapted from (H. Zhang et al., 2019))

The initial step toward self-attention is to break down each input feature into three separate vectors. $f(x)$, $g(x)$, and $h(x)$. To limit the number of channels, $1x1$ convolution is employed. Instead of inspecting every pixel, the self-attention component is concerned with the input activation's local regions. So, $h(x)$ is a representation of input features with a smaller number of channels and activation maps. The significance of the features of the self-attention map is determined using $f(x)$ and $g(x)$. The vector $g(x)$ at location x is calculated and compared with the vector $f(x)$ at all locations to compute an output feature at location x. The attention map is calculated with matrix multiplication between $f(x)$ and $g(x)$. Finally, the matrix multiplication

between $h(x)$ and the attention map gives a self-attention feature map. The self-attention layer enables the generator to generate images with fine details in every location that are accurately coordinated with fine characteristics in distant regions of the image.

To solve the challenge of limited convolution kernel size, (Yuan et al., 2020) added a self-attention mechanism into the generator's upsampling block to generate long-range dependence of the picture. As a result of this discovery, self-attention was added into the generator's decoder block to construct the image's long-term dependency.

4.3.2. Channel-Attention Mechanism

Since each filter in the Conv layer works with a nearby receptive field, the output after convolution cannot leverage contextual information outside of the adjacent region. To make the generator network focus on crucial features, (Yulun Zhang et al., 2018) proposed channel attention (CA) mechanism that makes use of feature channel interdependence. Global average pooling is utilized to convert channel-wise global location information into a channel descriptor.

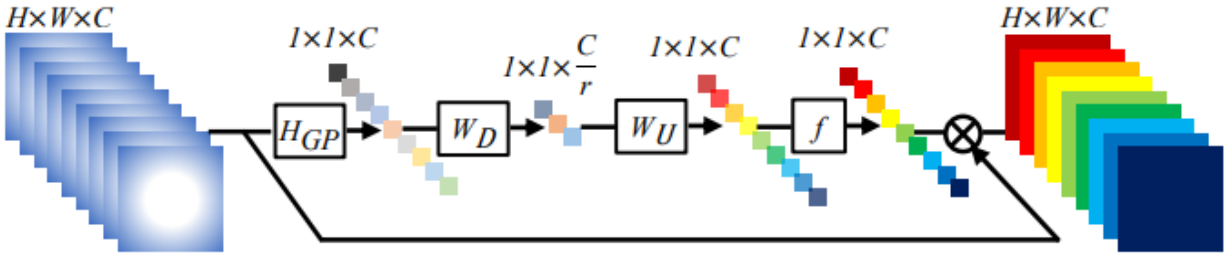


Figure 4. 4 Channel attention

(Source: adapted from (Yulun Zhang et al., 2018))

Let $Img = [img1, \dots, imgc, \dots, imgC]$ be an input containing C feature maps of size HXW.

which has C feature maps with the size of HXW. Img can be shrunk through spatial dimensions HXW to acquire the channel-wise statistics $z\ \mathcal{E}\ \mathbb{R}^c$.

Then the c^{th} element of z is determined by

$$z_c = H_{GP}(x_c) = \frac{1}{HXW} \sum_{i=1}^{H} \sum_{J=1}^{W} x_c(i,j) \qquad eq.(4.5)$$

Where $x_c\ (i,j)$ is the value at position (i,j) of c^{th} feature x_c. The global pooling function is denoted by $H_{GP}(.)$. Such channel statistics can be considered as a collection of the neighborhood descriptors, and it contributed to expressing the entire image.

A gating technique was also implemented to properly capture channel-wise dependencies from aggregated information via global average pooling. The gating mechanism should fulfill two conditions, as mentioned in (Hu et al., 2020): First and foremost, it should be capable of learning nonlinear interactions between channels. Second, because several channel-wise properties might be stressed rather than a solitary one-hot activation, it must learn a non-mutually exclusive connection. A basic gating system with a sigmoid function is used here

$$s = f\big(WU\delta(WDz)\big) \qquad eq.(4.6)$$

Where sigmoid gating and the ReLU function are denoted by $f(.)$ and $\delta(.)$ respectively. WD, is the weight of a Conv layer, that is used as a channel-downscaling with a reduction r ratio. The low-aspect signal is then expanded with ratio r by the channel-upscaling layer after ReLU activated it, whose weight is WU. The last channel statistics s are then obtained and employed to rescale the input x_c.

$$\widehat{x_c} = s_c \,.\, x_c \qquad eq.(4.7)$$

where s_c and x_c are the scaling factors and feature map in the c^{th} channel.

(Huang et al., 2019) utilized channel-wise attention at the decoder part in an MRI (Magnetic resonance imaging) reconstruction problem to focus on crucial features related to the ultimate objective while filtering out irrelevant and noisy features and achieves very promising outcomes in terms of common MRI reconstruction metrics which are SSIM, PSNR, and Normalized MSE. As a result of this discovery, Channel attention was employed in the generator's second, third, fourth, fifth, and sixth decoder layers to focus on significant features while filtering out irrelevant features.

4.4. Discriminator Network

The discriminator network is a patchGAN discriminator following the work (Isola et al., 2017) that penalizes structure only at the scale of nearby image patches. It uses Deep CNN to classify images as real or produced. Rather than learning to penalize structure across the entire image,

as the traditional GAN does, the patchGAN penalizes structure only at the scale of nearby image patches. It decreases the discriminator load greatly since distinguishing local patches requires far less model capacity than discriminating complete images. The discriminator network, as seen in the figure.4.5, includes convolution, instance normalization, and Leaky ReLU except for the first and last layers, the discriminator's first layer comprises convolution and instance normalization, while the last layer simply contains convolution. It contrasts the generated video frames to the ground truth frames.

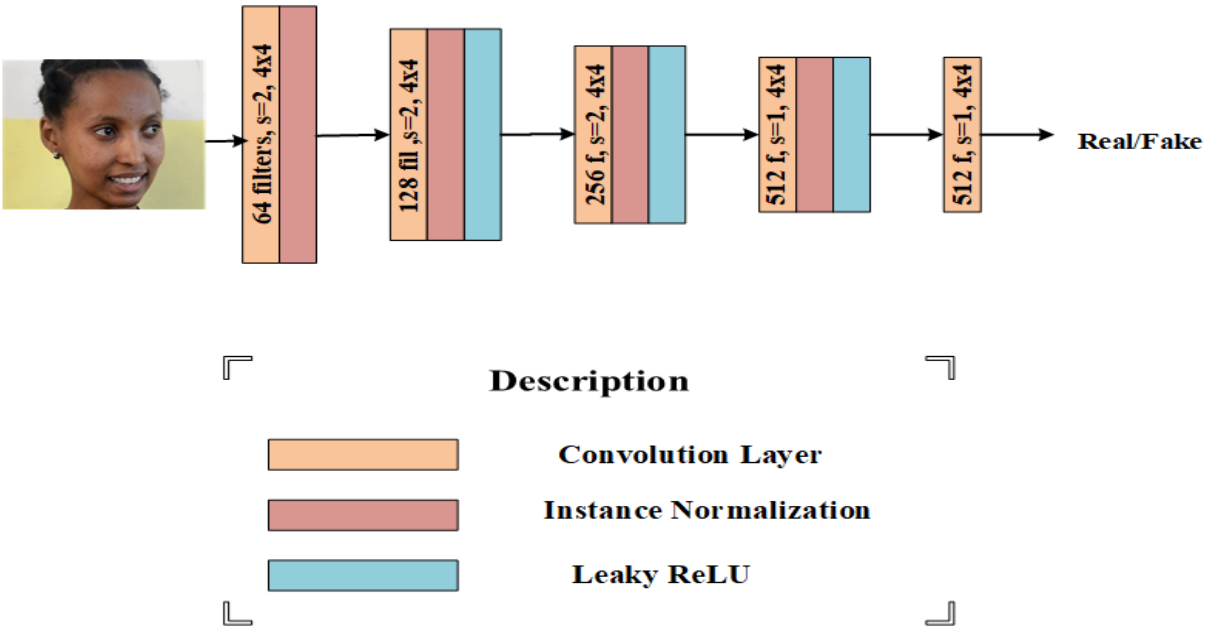


Figure 4. 5 Discriminator architecture

4.5. Model Learning Functions

The main aim of this work is to generate good-quality image sequences that are consistent in time. To achieve this objective different learning functions such as loss functions and attention mechanisms introduce to the network. The loss functions used in this work include adversarial loss $\mathcal{L}_a$ and reconstruction error $\mathcal{L}_{l1}$ between the produced and real image.

$$\mathcal{L}_t = \mathcal{L}_a + \mathcal{L}_{l1} \qquad eq.(4.8)$$

4.5.1. Adversarial loss

In this study, Mean Absolute Error is utilized to train the generator and the discriminator rather than vanilla GAN. It is determined by averaging the absolute difference between the actual and produced images.

$$\mathcal{L}_a = \frac{1}{2}|D(G(Io, e)) - D(It)| \qquad eq.(4.9)$$

Where D is the discriminator network that contrasts the generated image by the generator G and the ground truth image It.

4.5.2. Reconstruction loss

L1 is a typical loss function that minimizes the absolute disparities between the estimated values and the existing target values. The L1 loss function is more resilient and, in general, is unaffected by outliers. $l1$ loss utilized between the produced and the real images following (Isola et al., 2017).

$$\mathcal{L}_{l1} = |It - \widetilde{It}| \qquad eq.(4.10)$$

Where It is the ground truth (real) image and $\widetilde{It}$ is the generated image

4.7. Training Pseudocode

This section explains the overall training of the AI2VFEGAN model. At each epoch of the training, the neutral face images are taken sequentially. Then, the identity feature is encoded by the basic encoder, and the expression feature is encoded by the residual encoder. Next, the target feature is got from the basic feature, the residual feature, and the expression intensity value as stated in section 4.3. The target feature is fed to the decoder to synthesize the target expression image. Finally, the overall loss is calculated and backpropagation is performed.

Algorithm 1: Overall training algorithm of the AI2VFEGAN model

start = 0

end = 1

space = (end – start) / (num_frames – 1)

e = []

For each i, where i < num_frames:

    ei = start + i * space

    add ei to e

End For

$Input: Io, ei$

Where Io is input neutral face image and ei expression intensity value of one frame

$Output$: $\widetilde{It}_{ei}$

$Ground\ Truth$: It_{ei}

For the number of training epochs do:

$Train\ D$:

$\widetilde{It}_{ei} = G(Io, ei)$

$\boldsymbol{\mathcal{La}} = \frac{\mathbf{1}}{\mathbf{2}} \left|\boldsymbol{D}\left(\boldsymbol{\widetilde{It}_{ei}}\right) - \boldsymbol{D}(\boldsymbol{It_{ei}})\right|,$

Where $\widetilde{It}_{ei}$ is generated image at ei and It_{ei} ground truth image at ei

$update\ discriminator\ parameter\ to\ improve\ classification\ loss$

$Train\ G$:

$fb = Enc\_b(Io)$

$fr = Enc\_r(Io)$

$ft = fb + fr * ei$

$\widetilde{It}_{ei} = Dec(ft)$

$\boldsymbol{\mathcal{La}} = \boldsymbol{D}\left(\boldsymbol{\widetilde{It}_{ei}}\right)$

$\mathcal{L}l1 = |It_{ei} - \widetilde{It}_{ei}|$

$\mathcal{L}total = \mathcal{L}a + \mathcal{L}l1$

$update\ generator\ parameter\ to\ minimize\ reconstruction\ loss$

End for

The bold indicates the modified components

CHAPTER FIVE

IMPLEMENTATION OF THE PROPOSED MODEL

5.1. Chapter Overview

This chapter describes the implementation of the proposed model, the working environment, the data augmentation implementation, Attention-based image-to-video translation for synthesizing facial expression using GAN implementation, and the experiment class conducted are discussed.

5.2. Working Environment

This section discusses the environmental setup of hardware tools used for implementing the model.

Desktop:

- Processor: Intel®Intel™ i7-8700 CPU @ 20GHz 3.19 GHz
- Installed RAM: 8.00 GB
- Operating system: Window 10 pro
- System type: 64-bit operating system, x64-based processor

Desktop:

- Processor: Intel® Core™ i5-4590 CPU @ 3.30 GHz 3.30 GHz
- Installed RAM: 14.00 GB
- Operating system: Window 10 pro
- System type: 64-bit operating system, x64-based processor
- Graphics: Intel ® HD Graphics 4600
- GPU: GeForce RTX 2070 Super 6 GB RAM

5.3. Environmental Setup

- Anaconda: is an open-source tool that makes package administration easier. It is utilized to set up Python and its various modules. Anaconda version 1.9.7 is utilized to set up necessary libraries and modules.
- Jupyter notebook: an open-source, tool that integrates software code, explanatory text, computational output, and multimedia resources into a single page.

- Colab: is a free notebook that runs on the cloud and makes use of a free GPU. It lets you develop and run Python code and incorporates TensorFlow, OpenCV, Keras, and PyTorch.

5.4. Data Augmentation Implementation

Since the original datasets are too small, the data augmentation technique is employed to make the dataset larger. Image augmentation techniques can produce different variants of the images. From different image augmentation techniques: random flip, random brightness, random saturation, random contrast, and adjusted brightness method were used.

```
tf.random.set_seed(1)
bright_image = tf.image.random_brightness(image,
max_delta=32.0/255.0)
sat_image = tf.image.random_saturation(image, lower=0.5, upper=1.5)
con_image = tf.image.random_contrast(image, lower=0.5, upper=1.5)
bright_image2 = tf.image.adjust_brightness(image, 0.3)
flip_image = tf.image.random_flip_left_right(image, seed = 1)
```

Sample code 5. 1 Data augmentation implementation

5.5. Proposed Model Implementation

The proposed model contains one generator and one discriminator network. The generator network, as explained in section 4.3, is a U-Net generator that consists of two encoders and the decoder. Both the encoders and the decoder are seven-layer neural networks with a skip connection as (Weng & Zhu, 2021). The generator network's encoders both accept input images with 256x256x3 resolution. Tables 5.1 and 5.2 describe the layers of the generator and the discriminator architecture.

```
Def define_G(input_nc, output_nc, ngf, netG, norm='instance',
init_type='normal', init_gain=0.02, gpu_ids=[]):
    norm_layer = get_norm_layer(norm_type=norm)
    netG = UnetGenerator(input_nc, output_nc, 7, ngf,
norm_layer=norm_layer, gpu_ids=gpu_ids)

    return init_net(net, init_type, init_gain, gpu_ids)
```

Sample code 5. 2 Generator function implementation

The generator network's encoders both accept input images as input and the basic encoder hold a feature to retain the identity of the input subject and the residual encoder holds a feature to direct the expression shift from neutral to the peak.

```
f_t0 = self.netG(self.input_A, 1.0, 0.0, self.isTrain)

f_t0_res = self.netG(self.input_A, 0.0, 1.0, self.isTrain)
```

As discussed in section 4. expression intensity e is used to control expression change and is inferred by evenly distributing values within the range [0,1] according to the number of frames in the training dataset. This is achieved by using the NumPy linspace function. In essence, you provide the beginning and ending point for an interval, and also the number of breakpoints you desire inside that interval. The beginning and ending points are set to 0 and 1 respectively and the interval is the number of frames in a given video.

```
Intensity_value = np.linspace(0, 1, len(img_names))

alpha = intensity_value.tolist()
```

Sample code 5. 3 Expression intensity value implementation

The generator then generates expression images with different intensities by accepting the neutral face image and expression intensity value as input.

```
Fake_B = self.netG(self.input_A, 1.0, float(alpha), self.isTrain)
```

Then the adversarial loss is computed to assess whether the generated image is genuine or fake. The detailed implementation is discussed in appendix D.

Table 5. 1 Content of the generator architecture

Generator	
Layers in the encoders	Content of each block
1	Convolution – (filters=64, kernel=(4x4), strides=2)
2	Convolution – (filters=128, kernel=(4x4), strides=2), Instance normalization, LeakyReLU

3	Convolution – (filters=256, kernel=(4x4), strides=2), Instance normalization, LeakyReLU
4-6	Convolution – (filters=512, kernel=(4x4), strides=2), Instance normalization, LeakyReLU
7	Convolution – (filters=512, kernel=(4x4), strides=2), LeakyReLU
Layers in the decoder	
1	Deconvolution – (filters=64, kernel=(4x4), strides=2), tanh
2	Deconvolution – (filters=128, kernel=(4x4), strides=2), ReLU, instance normalization
3	Deconvolution – (filters=256, kernel=(4x4), strides=2), ReLU, instance normalization
4-6	Deconvolution – (filters=512, kernel=(4x4), strides=2), ReLU, instance normalization
7	Deconvolution – (filters=512, kernel=(4x4), strides=2), ReLU

Table 5. 2 Contents of the discriminator architecture

Discriminator	
Layers	Content
1	Convolution – (filters=64, kernel=(4x4), strides=2), Instance normalization, LeakyReLU
2	Convolution – (filters=128, kernel=(4x4), strides=2), Instance normalization, LeakyReLU
3	Convolution – (filters=256, kernel=(4x4), strides=2), Instance normalization, LeakyReLU
4	Convolution – (filters=512, kernel=(4x4), strides=2), Instance normalization, LeakyReLU

5.5.1. Self-attention Implementation

As discussed in section 4.3.1 a self-attention layer helps the generator network to create images with fine details in every location that are precisely coordinated with fine features in distant parts of the image. The self-attention layer is integrated at the first decoder layer of the generator and it accepts the feature from the last convolution.

```
Class Self_Attn(nn.Module):
    def __init__(self, in_channels):
        super(Self_Attn, self).__init__()
        self.in_channels = in_channels
        self.snconv1x1_theta = snconv2d(in_channels=in_channels,
out_channels=in_channels//8, kernel_size=1, stride=1, padding=0)
        self.snconv1x1_phi = snconv2d(in_channels=in_channels,
out_channels=in_channels//8, kernel_size=1, stride=1, padding=0)
        self.snconv1x1_g = snconv2d(in_channels=in_channels,
out_channels=in_channels//2, kernel_size=1, stride=1, padding=0)
        self.snconv1x1_attn = snconv2d(in_channels=in_channels//2,
out_channels=in_channels, kernel_size=1, stride=1, padding=0)
        self.maxpool = nn.MaxPool2d(2, stride=2, padding=0)
        self.softmax  = nn.Softmax(dim=-1)
        self.sigma = nn.Parameter(torch.zeros(1))
```

Sample code 5. 4 Self-attention implementation

5.5.2. Channel Attention Implementation

Channel attention is also integrated into the generator network to help the generator network to focus on crucial features. It accepts two inputs the feature from the previous convolution and the reduction value, the reduction value is 16.

```
Class ChannelAttention(nn.Module):
    def __init__(self, num_features, reduction):
        super(ChannelAttention, self).__init__()
        self.module = nn.Sequential(
            nn.AdaptiveAvgPool2d(1),
            nn.Conv2d(num_features, num_features // reduction,
kernel_size=1),
            nn.ReLU(inplace=True),
            nn.Conv2d(num_features // reduction, num_features,
kernel_size=1),
            nn.Sigmoid()
        ))
```

Sample code 5. 5 Channel attention implementation

5.5.3. Discriminator Network Implementation

The discriminator used in this work is the patchGAN discriminator following the pix2pix (Isola et al., 2017) work. The patchGAN discriminator utilizes deep CNN to classify images as genuine or fake. Instead of the entire image like the traditional GAN, the patchGAN penalizes structure only at the scale of nearby image patches. It significantly reduces the discriminator burden because discriminating local patches requires significantly less model capacity than discriminating entire images.

```
Class PatchDiscriminator(nn.Module):
    def __init__(self, input_nc, ndf=64, n_layers=3,
norm_layer=nn.BatchNorm2d, use_sigmoid=False):
        super(PatchDiscriminator, self).__init__()
        if type(norm_layer) == functools.partial:
            use_bias = norm_layer.func == nn.InstanceNorm2d
        else:
            use_bias = norm_layer == nn.InstanceNorm2d
        sequence = [
            nn.Conv2d(input_nc, ndf, kernel_size=4, stride=2,
padding=1),nn.LeakyReLU(0.2, True)]
        nf_mult = 1
        nf_mult_prev = 1
```

```
        for n in range(1, n_layers):
            nf_mult_prev = nf_mult
            nf_mult = min(2 ** n, 8)
            sequence += [
                nn.Conv2d(ndf * nf_mult_prev, ndf * nf_mult,
                          kernel_size=4, stride=2, padding=1,
bias=use_bias),norm_layer(ndf * nf_mult),nn.LeakyReLU(0.2, True)]
        nf_mult_prev = nf_mult
        nf_mult = min(2 ** n_layers, 8)
        sequence += [
            nn.Conv2d(ndf * nf_mult_prev, ndf * nf_mult,
                      kernel_size=4, stride=1, padding=1,
bias=use_bias), norm_layer(ndf * nf_mult), nn.LeakyReLU(0.2, True)]
        sequence += [nn.Conv2d(ndf * nf_mult, 1, kernel_size=4,
stride=1, padding=1)]
        if use_sigmoid:
            sequence += [nn.Sigmoid()]
        self.model = nn.Sequential(*sequence)
    def forward(self, input):
        return self.model(input)
```

Sample code 5. 6 Discriminator network implementation

5.6. Hyperparameter Configuration

A hyperparameter is a parameter that is established before the start of the learning process, such as batch size, number of epochs, optimization, learning rate, and loss weight. These factors are adjustable and have a direct impact on how successfully a model trains. Optimizers are techniques or approaches that are used to adjust the characteristics of the neural network, such as weights and learning rate, to minimize losses. Adam optimizer with learning-rate=0.002, beta_1=0.5, and beta_2=0.999 was utilized to update the network weight. Adam is described as integrating the benefits of two other extensions of stochastic gradient descent. Specifically: AdaGrad and RMSProp (Kingma & Ba, 2015), and it is straightforward to implement, computationally efficient, and requires little memory. The step size, often known as the "learning rate," is the amount by which the weights are changed during training. The learning rate and the weight loss for l1 used for this work are similar to (G. Shen et al., 2019). The model

is trained with a batch size of one for 1000 epochs. Stride=2 is used for feature downsampling and kernel size=4 is used to avoid checkerboard-like artifacts in the generated images that might happen when utilizing transpose convolutional layer (Cheong, n.d.).

Table 5. 3 Hyperparameter configuration

Hyperparameter	Value
Batch size	1
Number of epochs	1000
Optimizer	Adam
Learning rate	0.0002
Loss weight for l1	100
Stride	2
Kernel size	4

5.7. Experiment Class

To improve the image-to-video translation for synthesizing facial expressions performing different experiments and testing them is necessary. Five different experiments are conducted as shown below in table 5.4. The first experiment is the implementation of the baseline work. As stated in section 3.4 AffineGAN infer expression intensity value by applying affine transformation in the latent space and it used alpha discriminator to control the value from deviating away from the range [0,1]. BCE loss is used to train the generator and the discriminators.

The second experiment is the initial work of the proposed model. Expression intensity value inferred as stated in sections 4.1 and 5.5. Sine the NumPy linespace function bound the values to the range [0,1], the alpha discriminator is not necessary like AffineGAN. For the local discriminator, the boundary image is used as an informative region to calculate the loss between the multiplication of the informative region and the ground truth image with the multiplication of the informative region with a generated image. Instead of using the mouth region only as an informative region like (Fan et al., 2019; G. Shen et al., 2019), the boundary image is used as an informative region since the boundary image can hold the structure of the face image and is also better at describing the facial pose than only the mouth area. MAE loss is used to train the generator and the discriminators. The boundary is detected using an open-source face alignment

library(Bulat & Tzimiropoulos, n.d.). In the third experiment, the self-attention mechanism was added at the first decoder layer of the generator for good-quality image generation. In the fourth experiment, the channel attention was added at the second, third, fourth, fifth, and sixth layers of the decoder. In the last experiment, the model is trained without the local discriminator to show the effect it has on the model. This experiment is the overall proposed model.

Table 5. 4 List of experiment class

Notation	Experiment	Dataset used
AffineGAN	Baseline work	MUG and Local facial expression dataset
MAEGAN	Expression intensity inferred as stated in section 4.1, boundary image used as an informative region for the local discriminator and MAE loss used to train the generator and the discriminators	MUG and Local facial expression dataset
MAEGAN+SA	With the addition of self-attention layer and with local discriminator	MUG and Local facial expression dataset
MAEGAN+SA+CA	With the addition of both self and channel attention and with local discriminator	MUG and Local facial expression dataset
AI2VFEGAN	With the addition of both self and channel attention but without the local discriminator	MUG and Local facial expression dataset

CHAPTER SIX

RESULTS AND DISCUSSIONS

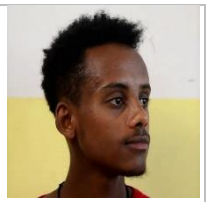
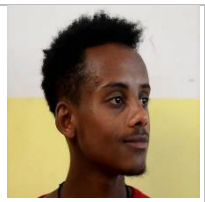
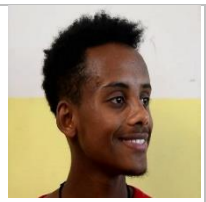
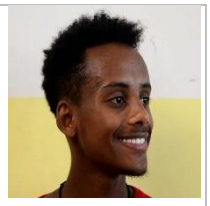
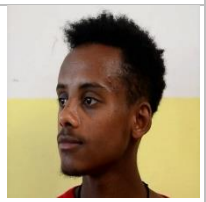
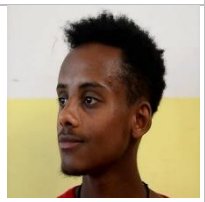
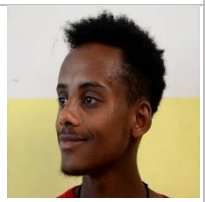
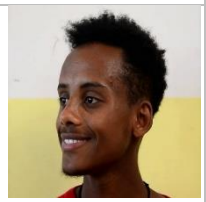
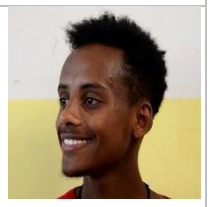
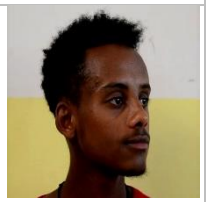
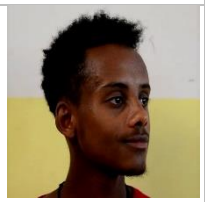
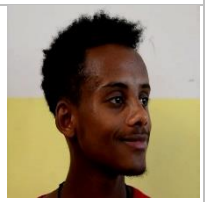
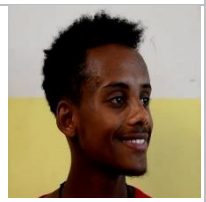
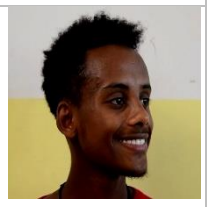
6.1. Chapter Overview

This chapter discusses the outcomes of the data augmentation, the baseline work, and the different experiments carried out including the proposed solution, and gives a comparative description of the findings with figures, graphs, and tables.

6.2. Results of the Study

6.2.1. Data Augmentation results

Using image data augmenting, a training set can be made larger by generating different versions of the images. Figures 6.1 and 6.2 shows data augmentation results of both datasets using horizontal left to right flip, random brightness with brightness factor 32/255, random saturation with saturation range [0.5,1.5], random contrast with contrast range [0.5,1.5], and adjusted brightness with brightness factor 0.3 techniques.

Original Image					
Horizontal Flip					
Random Brightness					

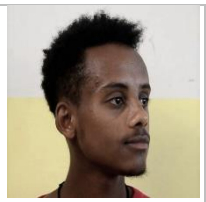
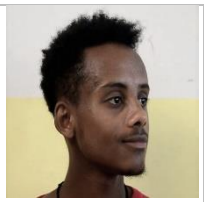
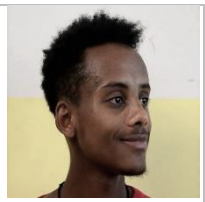
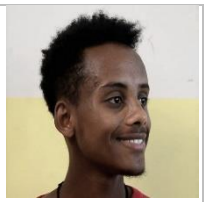
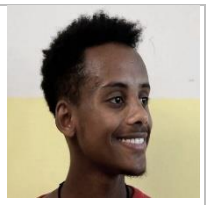
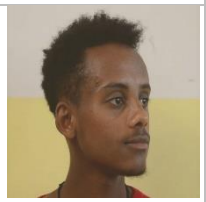
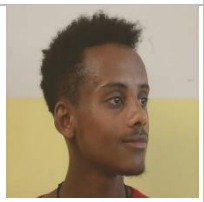
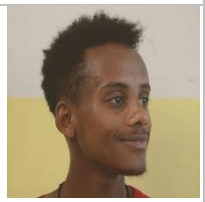
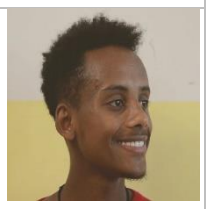
Random Saturation					
Random Contrast					
Adjusted brightness					

Figure 6. 1 Sample augmentation result on Local dataset

6.2.2. Experimental Results

For this study, five experiments were carried out to get better results, and all the experiments were done on the same dataset, hardware, and software configuration for comparison purposes. The results are shown for the two datasets that are MUG dataset and the Local dataset.

6.2.2.1. AffineGAN

AffineGAN contains one generator and three discriminators. The first discriminator is used to contrast generated image with the ground truth (real) image. The second discriminator is used to contrast the multiplication of the informative region (mouth region) and the produced image with the multiplication of the informative region and ground truth image. Mouth region mask used as the informative region for the local discriminator. The third discriminator is used to keep the expression intensity value within the range [0,1]. BCE loss is used to train the generator and the discriminators.

Input image	e = 0	e = 0.142	e = 0.285	e = 0.428	e = 0.571	e = 0.714	e = 0.857	e = 1
								

Figure 6. 2 Result of AffineGAN

The first row is the expression intensity value of eight frames, the second and the third row shows results of AffineGAN on the MUG dataset and Local dataset respectively.

As shown in figure 6.3, in the first row there is a sudden change at expression intensity value of 0.142 which might affect the consistency of the video. This shows that there is a probability that inconsistency might occur in the AffineGAN network.

There is also a quality problem at expression intensity values 0.857, and 1 on the MUG dataset. On the local dataset, there is an issue with the mouth and teeth part.

Figure 6. 3 Generated frames with AffineGAN network

6.2.2.2. MAEGAN

To solve the above-stated problems the second experiment was carried out. In this experiment, expression intensity value inferred as stated in sections 4.3 and 5.5, to make the generator generate consistent image sequences as well as different size videos while remaining in the range [0,1]. The alpha discriminator is not necessary for this case since the NumPy linesapce function bound the value to the range [0,1] by itself. For the local discriminator, the boundary image is used as an informative region to calculate the loss between the multiplication of the informative region and the real image with the multiplication of the informative region with a produced image. Instead of using the mouth region only as an informative region like (Fan et al., 2019; G. Shen et al., 2019), the boundary image is used as an informative region since the boundary image can hold the structure of the face image and is also better at describing the facial pose than only the mouth area. MAE loss is utilized to train the generator and the discriminators.

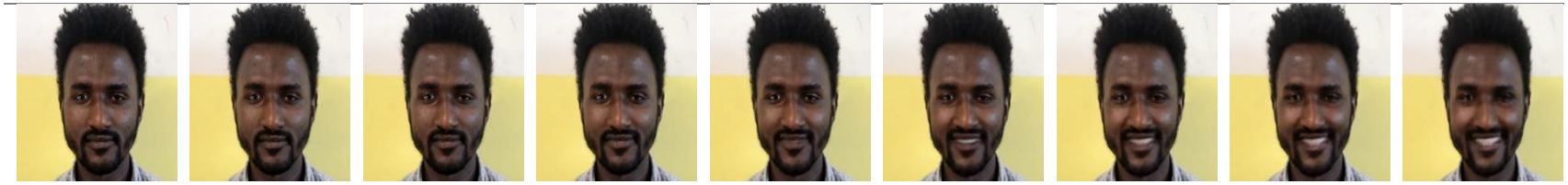
Input image	**e = 0**	**e = 0.142**	**e = 0.285**	**e = 0.428**	**e = 0.571**	**e = 0.714**	**e = 0.857**	**e = 1**
								

Figure 6. 4 Result of MAEGAN

The first row is the expression intensity value of eight frames, the second and the third row shows results of MAEGAN on the Local dataset.

As shown in figure 6.6, there are no sudden changes in performing the expression, and the quality of the image sequence also gets better compared to the previous one. But generated images are somehow blurry, especially at peak expression intensity value.

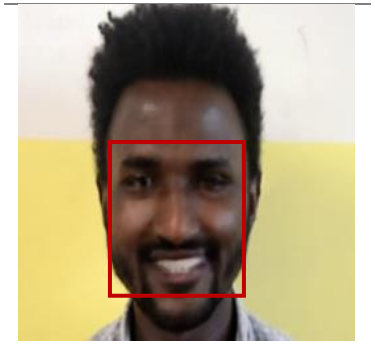


Figure 6. 5 Generated frame with MAEGAN

6.2.2.3. MAEGAN+SA

To solve the blurry problem in the previous experiment self-attention layer was added at the decoder of the generator.

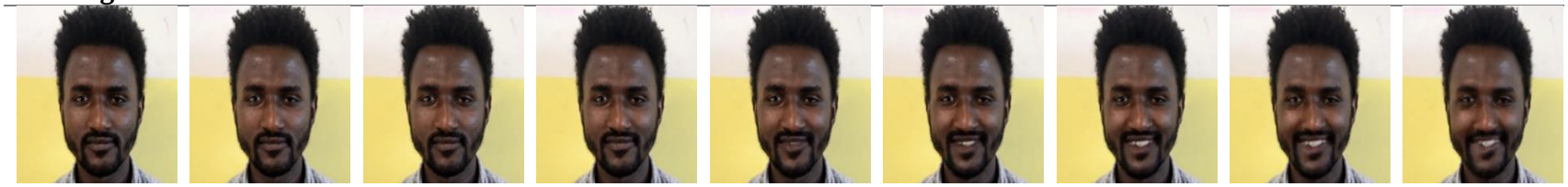
Input image	e = 0	e = 0.142	e = 0.285	e = 0.428	e = 0.571	e = 0.714	e = 0.857	e = 1
								

Figure 6. 6 Result of MAEGAN+SA

The first row is the expression intensity value of eight frames, the second and the third row shows results of MAEGAN+SA on the Local dataset.

As shown in figure 6.8, the blurry decreases compared to the previous experiment but there are still some artifacts at the generated image images at peak expression intensity values.

Figure 6. 7 Generated frames with MAEGAN+SA

6.2.2.4. MAEGAN+SA+CA

To solve the quality issue the channel attention was added at the second, third, fourth, fifth, and sixth layers of the decoder.

Input image	e = 0	e = 0.142	e = 0.285	e = 0.428	e = 0.571	e = 0.714	e = 0.857	e = 1

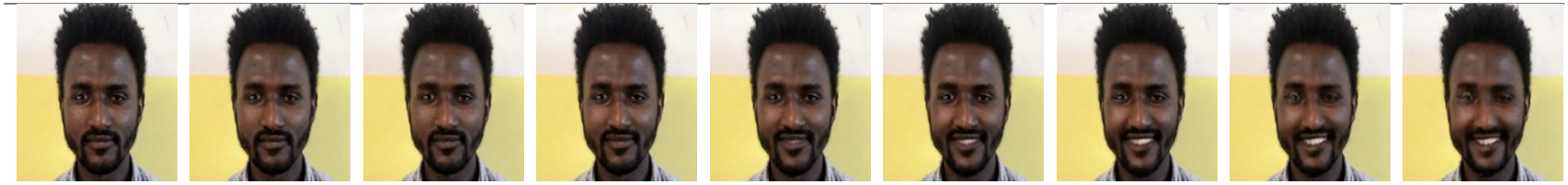


Figure 6. 8 Result of MAEGAN+SA+CA

The first row is the expression intensity value of eight frames, the second and the third row shows results of MAEGAN+SA+CA on the Local dataset.

As shown in figure 6.10, the image sequences have good quality compared to the previous one but there are some artifacts on the images.

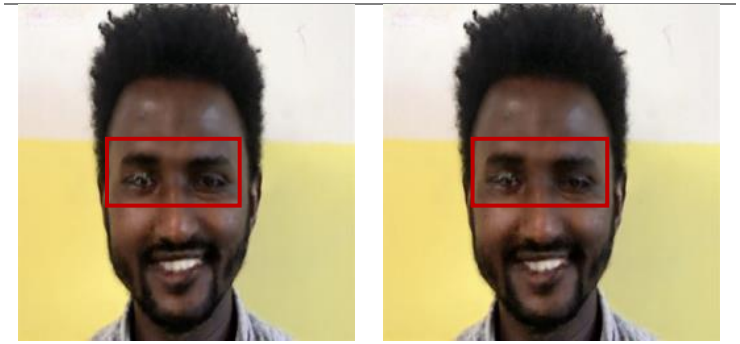


Figure 6. 9 Generated frames with MAEGAN+SA+CA

6.2.2.5. AI2VFEGAN

In this experiment, the network is trained without the Local discriminator. Since the Local discriminator loss remains constant for almost all the epochs on the previous experiments, the model trained without the local discriminator to show the effect it has on the network.

As shown in figure 6.13, the image sequences have good quality for both datasets compared to the previous experiments. The result shows that the Local discriminator is not that necessary to the model to generate quality image sequences.

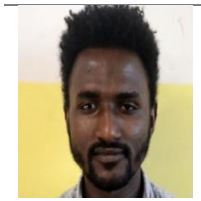
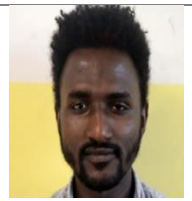
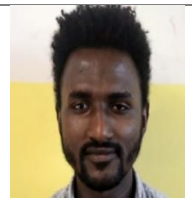
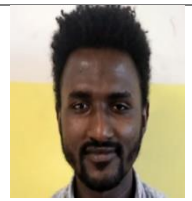
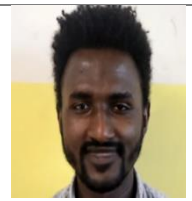
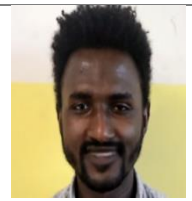
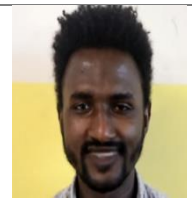
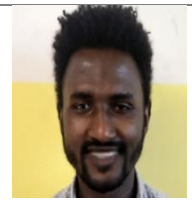
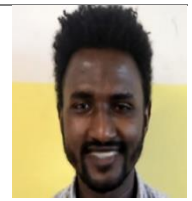
Input image	**e = 0**	**e = 0.142**	**e = 0.285**	**e = 0.428**	**e = 0.571**	**e = 0.714**	**e = 0.857**	**e = 1**
								

Figure 6. 10 Result of AI2VFEGAN

The first row is the expression intensity value of eight frames, the second and the third row shows results of AI2VFEGAN on the Local dataset.

Figure 6. 11 Generated frames with AI2VFEGAN

The generated frame at expression intensity values 0.857 and 1 by AI2VFEGAN**,** shows the quality improvement from the previous experiments on both datasets

Figure 6.15 shows the generated image sequences for the four facial expressions (happy, anger, surprise, and disgust) with four different sides on the Local dataset.

Input image	**e = 0**	**e = 0.142**	**e = 0.285**	**e = 0.428**	**e = 0.571**	**e = 0.714**	**e = 0.857**	**e = 1**
								

Figure 6. 12 Generated frames of four catagories on the Local dataset

The first row is the expression intensity value of eight frames, the second row is the produced frames with happy expression with near 45-degree left position, the second row shows the produced frames with anger expression with near 90-degree left position, the third row shows the produced frames with surprise expression with near 45-degree right position, the last row shows the produced frames with disgust expression with near 90-degree right position with the AI2VFEGAN model on Local dataset.

Figure 6.16 shows the generated image sequences for the anger expression with frontal and two sides of the left and right side on the Local dataset.

Input image	e = 0	e = 0.142	e = 0.285	e = 0.428	e = 0.571	e = 0.714	e = 0.857	e = 1

Figure 6. 13 Result of anger expression on the Local dataset

The first row is the expression intensity value of eight frames, the second row is the produced frames with anger expression with near 45-degree left position, the second row shows the produced frames with anger expression with the near 45-degree right position, the third row shows the produced frames with anger expression with near 90-degree left position, the last row shows the produced frames with anger expression with near 90-degree right position with the AI2VFEGAN model on Local dataset.

The generator is also able to generate a video with different frame sizes while remaining within the range [0,1]. Figures. 6.17. and 6.18 show the generated image sequences with frame sizes five and three respectively.

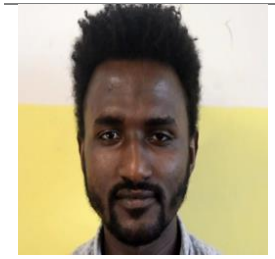
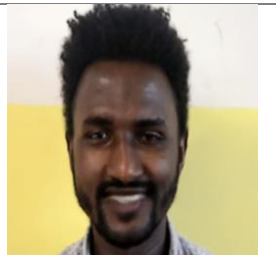
Input image	**e = 0**	**e = 0.25**	**e = 0.5**	**e = 0.75**	**e = 1**
					

Figure 6. 14 Generated image sequences with five frames

The first row is the expression intensity value of five frames, the second row is the produced frames of the AI2VFEGAN model with a happy expression on the Local facial expression dataset

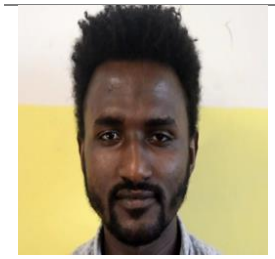
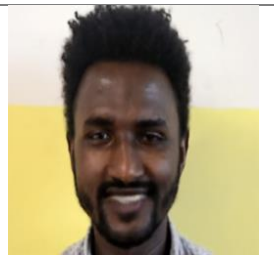
Input image	e = 0	e = 0.5	e = 1
			

Figure 6. 15 Generated image sequences with three frames

The first row is the expression intensity value of three frames, the second row is the produced frames of the AI2VFEGAN model with a happy expression on the Local facial expression dataset

6.3. Sample training log for AI2VFEGAN

Figure 6.19 shows the generator loss of AI2VFEGAN, at the first epoch the loss was around 9.8, and even if there is an up and down in the loss value, the loss decreases smoothly throughout the training epochs. At epoch 1000 the loss is around 2.17.

Figure 6.20 shows the discriminator loss of AI2VFEGAN, at the first epoch the loss was around 0.48. There is an up and down in the loss value and at epoch around 55 the loss value increases than the first epoch loss value but after 750 epochs the loss decreases. At epoch 1000 the loss is around 0.013.

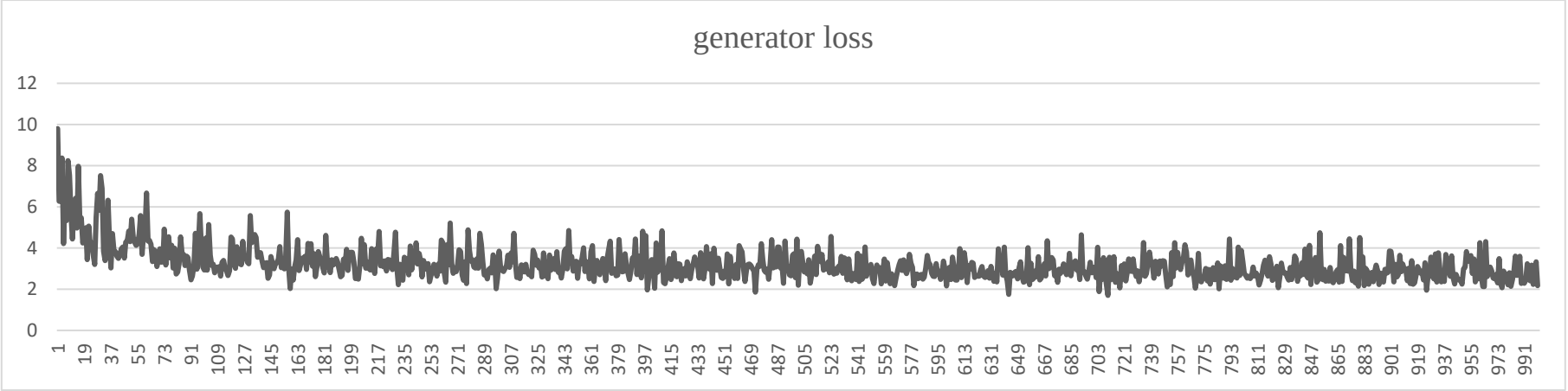


Figure 6. 16 Generator loss graph

The y-axis indicates the loss value and the x-axis indicates the epoch

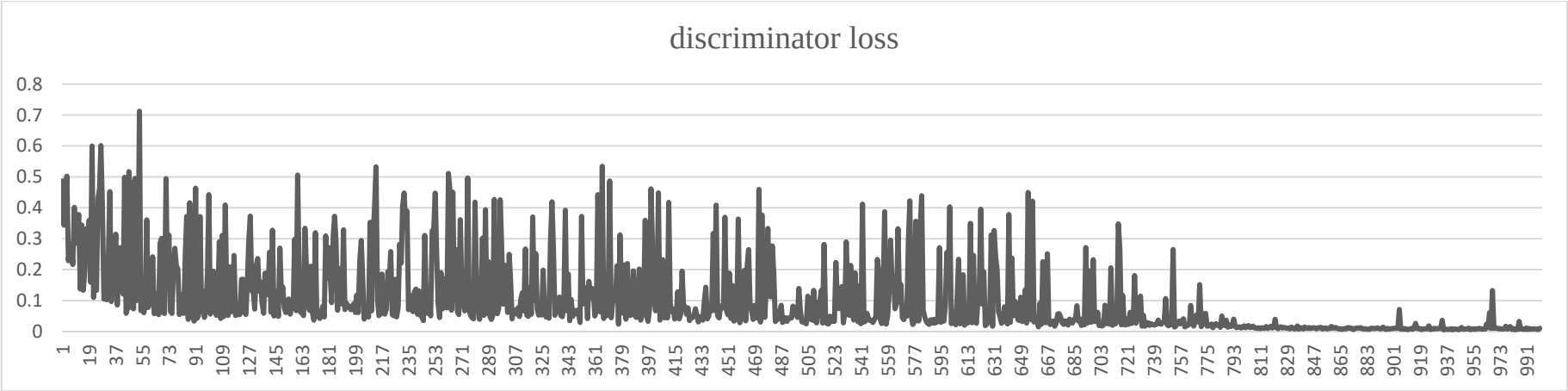


Figure 6. 17 Discriminator loss graph

The y-axis indicates the loss value and the x-axis indicates the epoch

6.4. Evaluation Metrics

6.4.1. Average Content Distance (ACD)

ACD-I, ACD-C, and ACD-G metrics were utilized to measure the model performance of the different experiments that were carried out. ACD-I is utilized to evaluate the quality of face identities and it computes the average distance between the original image and the output frames. ACD-C is used to assess content consistency and it computes the average distance between all possible pairs of frames in a video. ACD-G is used to evaluate expression changes it computes the average frame-to-frame distance between the generated frames and the corresponding ground-truth ones. Table 6.1 shows the ACD-I, ACD-C, and ACD-G scores of the five experiments. All the ACD scores were also computed for the real (ground truth) for reference. The ACD scores are lower, the better the model performance is.

Table 6. 1 ACD-I, ACD-C, and ACD-G scores of the five experiments

Number	Experiment	ACD-I	ACD-C	ACD-G
1	AffineGAN	1.606±0.018	1.452±0.008	1.769±0.007
2	MAEGAN	1.58±0.016	1.414±0.007	1.738±0.011
3	MAEGAN+SA	1.584±0.001	1.430±0.009	1.744±0.01
4	MAEGAN+SA+CA	**1.578±0.0017**	1.415±0.0015	1.737±0.009
5	AI2VFEGAN	1.5785±0.015	**1.413±0.008**	**1.733±0.014**
6	Ground truth	1.38±0.02	1.400±0.01	0

The bold value indicates the best result in the experiments

6.4.2. Structural Similarity Index and Peak Signal to Noise Ratio

Table 6.2 summarizes the trained model's reconstruction capabilities by utilizing PSNR and SSIM metrics to assess the similarity of the produced image to the original image. Higher PSNR and SSIM values indicate that the generated image has become close to the original image.

Table 6. 2 SSIM and PSNR score of the five experiments

Number	Experiment	SSIM	PSNR
1	AffineGAN	0.808	32.922
2	MAEGAN	0.833	33.408
3	MAEGAN+SA	0.837	33.414
4	MAEGAN+SA+CA	0.838	33.418
5	AI2VFEGAN	**0.842**	**33.574**

The bold value indicates the best result in the experiments

6.5. Results Discussion

6.5.1. Discussion on ACD scores result

To evaluate the model performance of all the experiments performed, ACD-I, ACD-C, and ACD-G metrics were calculated as shown in Table 6.1. Different experiments were carried out, the first experiment was the baseline work which scores ACD-I of 1.606±0.018. In the second experiment, MAEGAN employed MAE loss to train the generator and the discriminators, the expression intensity value inferred as mentioned in section 4.3, and the boundary image used as an informative region to compute the loss of the informative region. This model lowers the ACD-I from 1.606±0.018 to 1.58±0.016, indicating that the identity quality of the produced image sequences is higher than the prior experiment. In the third experiment, MAEGAN+SA introduced the self-attention layer into the generator network to generate high-quality images. This model scores ACD-I of 1.584±0.001, with a few variations from the prior experiment. The fourth experiment, MAEGAN+SA+CA, included the channel attention mechanism into the generator network to generate high-quality images. This model reduces the ACD-I score from 1.584±0.001 to 1.578±0.0017, indicating a significant increase in the identity quality of the produced image sequences over MAEGAN+SA. The AI2VFEGAN experiment had the ACD-I value of 1.5785±0.015 which is almost equal to the previous experiment. This indicates that the local discriminator does not help the network to generate good-quality images.

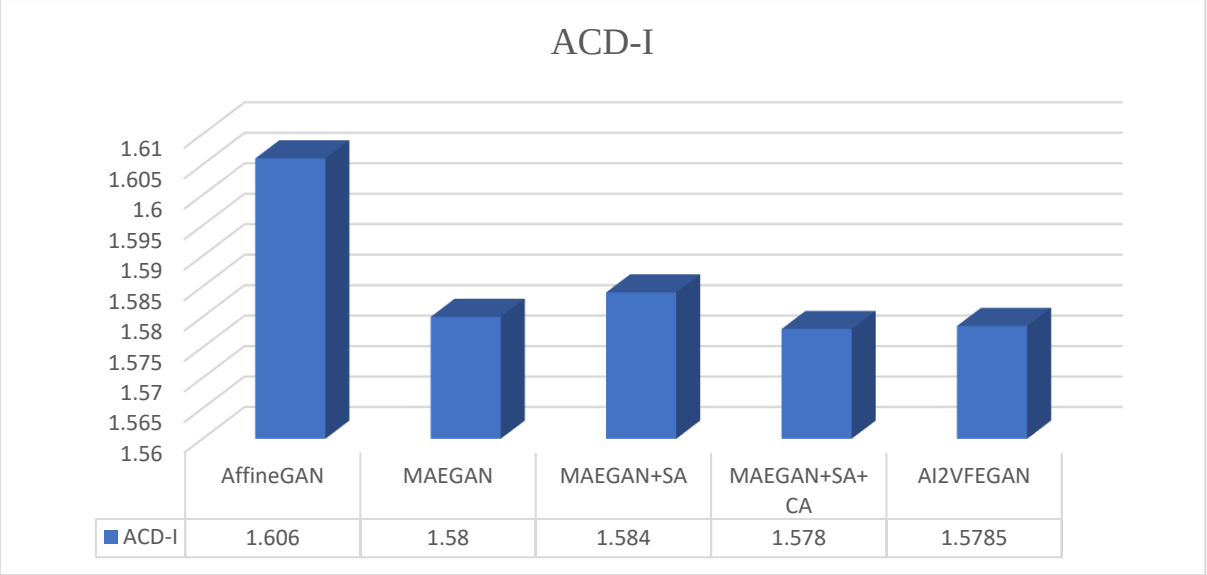


Figure 6. 18 ACD-I scores of the five experiments

The y-axis indicates the ACD-I score and the x-axis indicates the five experiment

The ACD-C score evaluates the consistency of the generated image sequences. The baseline work scores ACD-C of 1.452±0.008. MAEGAN experiment scores ACD-C of 1.414±0.007 which is lower than the previous experiment. Indicating that the MAEGAN outperforms the prior experiment in terms of creating consistent image sequences. Even though the expression intensity value is calculated in the same manner MAEGAN+SA scores ACD-C of 1.430±0.009, which is higher than the previous experiment this is due to some artifacts on the generated images with the MAEGAN+SA model, indicating that quality issues in the generated image sequences can impair the video's consistency. MAEGAN+SA+CA scores ACD-C of 1.415±0.0015, with a few variations from the MAEGAN due to the expression intensity value being inferred similarly. The AI2VFEGAN score an ACD-C of 1.413±0.008, which is the lowest of all the prior experiments and close to the ground truth ACD-C score this indicates the generated image sequences is consistent compared to the baseline work.

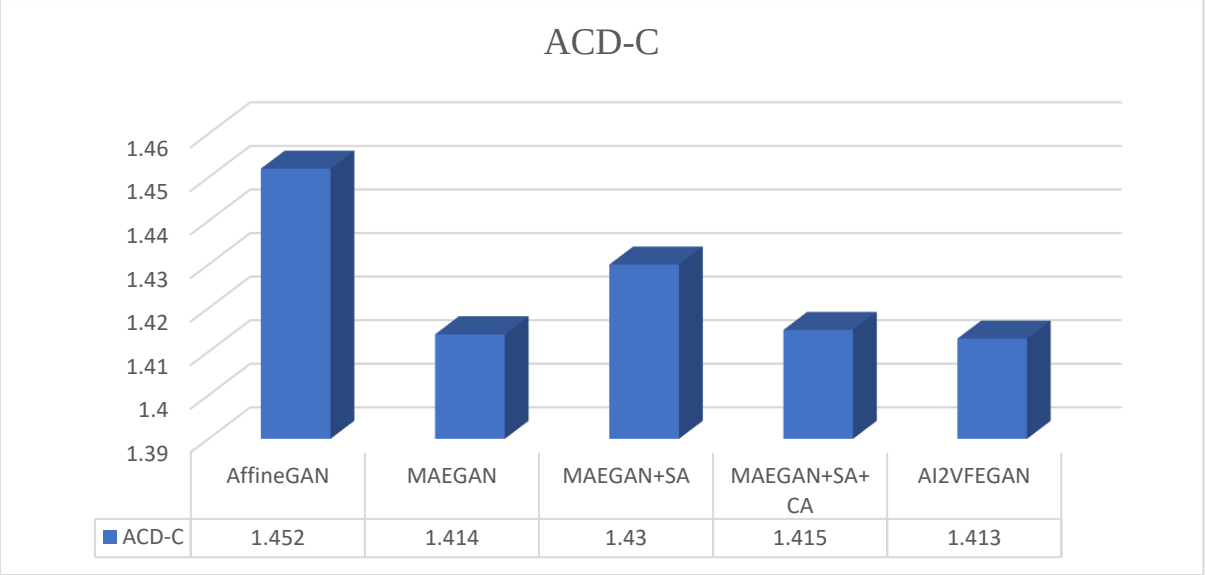


Figure 6. 19 ACD-C scores of the five experiments

The y-axis indicates the ACD-C score and the x-axis indicates the five experiment

ACD-G score was used to evaluate the expression change of the generated frames. The baseline work scores ACD-G score of 1.769±0.007. MAEGAN reduces ACD-G from 1.769±0.007 to 1.738±0.011. MAEGAN+SA scores ACD-G of 1.744±0.01 which is slightly higher than the MAEGAN, this indicates that the MAEGAN is better at performing expression than MAEGAN+SA. MAEGAN+SA+CA scores ACD-G of 1.737±0.009 which is better than the previous experiments. The proposed model scores the lowest ACD-G score, which is 1.733±0.014, from all experiments. This indicates that the last experiment is better at performing expression than all previous experiments.

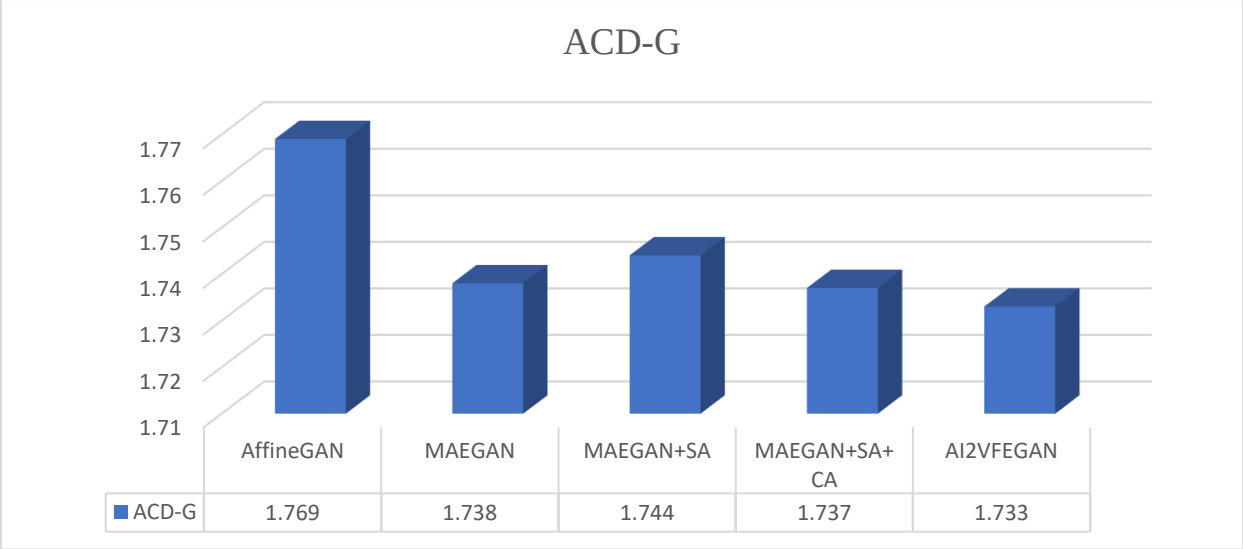


Figure 6. 20 ACD-G scores of the five experiments

The y-axis indicates the ACD-G score and the x-axis indicates the five experiment

6.5.2. Discussion on SSIM scores result

The SSIM evaluates the perceived quality of images and videos using three terms: brightness, contrast, and structure. As demonstrated in Table 6.2 the MAEGAN model scores 0.833 which is better than the baseline work. The MAEGAN+SA and MAEGAN+SA+CA score 0.837 and 0.838 respectively. The AI2VFEGAN improved the SSIM score from 0.838 to 0.842 indicating that the MAE loss, self-attention, and channel attention improve the quality of the generated image sequences and the local discriminator is not useful for the model.

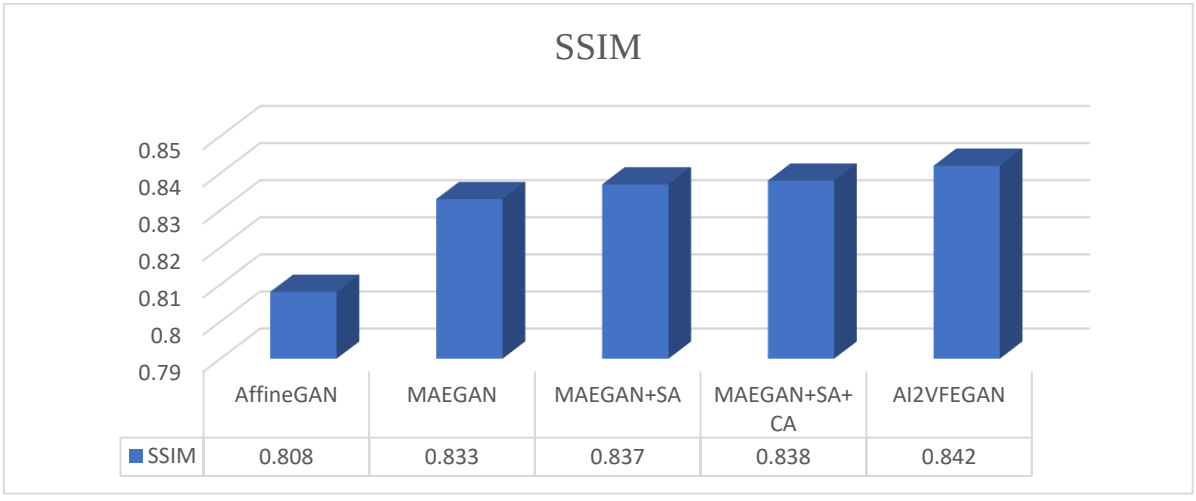


Figure 6. 21 SSIM score of the five models

The y-axis indicates the SSIM score and the x-axis indicates the five experiment

6.5.3. Discussion on PSNR scores result

PSNR is another metric used for evaluating the quality of the generated image sequences. As demonstrated in Table 6.2 the AffineGAN scores the lowest PSNR which is 32.922. The MAEGAN scores 33.408, indicating that the quality of the produced image sequences is higher than the prior experiment. MAEGAN+SA and MAEGAN+SA+CA score 33.414 and 33.418 respectively which is a bit higher than MAEGAN since they generate image sequences somehow sharp than the MAEGAN. The AI2VFEGAN model scores 33.574 which is the highest of all the other experiments, indicating that the quality of the image sequences generated by this model is better than previous experiments.

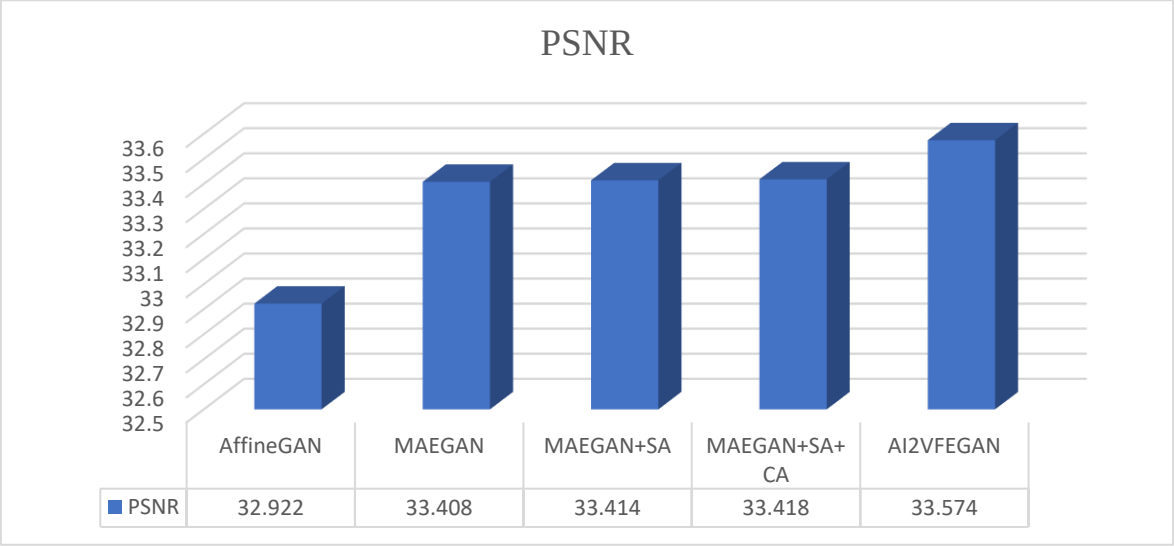


Figure 6. 22 PSNR score of the five models

The y-axis indicates the PSNR score and the x-axis indicates the five experiment

CONCLUSION AND FUTURE WORKS

Conclusion

Recently, with the achievement of Generative Adversarial Networks (GANs), significant progress has been achieved in image generation tasks, one of which is facial expression synthesis. Recent methods, that generate facial expression video from a single image, are only focused on frontal and near frontal facial expression synthesis. Another main issue in video generation is the consistency of image sequences, some methods assign expression intensity values manually to the range [0,1] to make the generated image sequence consistent in time. However, the manual annotation failed when the video is incomplete. Recent work assigns the expression intensity value automatically by applying affine transformation in the latent space. However, to determine the expression intensity value, this work must extract features of the target ground truth images and the quality of the generated image sequences decreases as the expression increases. So, this work proposed attention-based image-to-video translation for synthesizing facial expression using GAN to solve the stated problems.

This study introduced a method to infer the expression intensity value to make the generator able to generate consistent and varying frame size videos. This work also introduced self-attention and channel attention mechanisms to the network architecture and train the model with Mean Absolute Error loss. The local dataset was also prepared with frontal and with two different face positions on the left and right sides.

Different experiments were performed for comparison. In the first experiment, expression intensity value inferred by evenly distributing values to the range [0,1] according to the number of frames in a given video, the MAE loss utilized to train the model, and the boundary image used an informative region. This experiment shows an improvement in the quality of image sequences and consistency among them as shown in fig. 6.2.2.2. It improves the ACD-I score from 1.606±0.018 to 1.58±0.016, ACD-C score from 1.452±0.008 to 1.414±0.007, and ACD-G score from 1.769±0.007 to 1.738±0.011. And it also improves the SSIM score from 0.808 to 0.833 and PSNR from 32.922 to 33.408. In the second experiment, the self-attention layer is introduced into the generator network to generate good-quality images. This model tried to make the image sharp but there are some artifacts on the generated image sequences. It gave an ACD-I score of 1.584±0.016, ACD-C score of 1.43±0.009, and ACD-G score of 1.744±0.01, which

are a bit higher than the previous experiment. But it improves the SSIM score from 0.833 to 0.837 and PSNR score from 33.408 to 33.414. In the fourth experiment, the channel attention layer was introduced into the generator network for quality image generation. This model made a good improvement to the generated images as shown in fig 6.2.2.4. It improved the ACD-I score from 1.584±0.001 to 1.578±0.0017, ACD-C score from 1.43±0.009 to 1.415±0.0015, and ACD-G from 1.744±0.01 to 1.737±0.009. The SSIM and PSNR score also improved from 0.837 to 0.838 and from 33.414 to 33.418 respectively. The last experiment is the overall proposed model, it trained without the local discriminator. This model gave an ACD-I score of 1.5785±0.015 which is almost equal to the previous experiment, ACD-C score of 1.413±0.008, and ACD-G of 1.733±0.014 which is better than the other experiments. The SSIM and PSNR scores improved from 0.838 to 0.842 and 33.418 to 33.574 respectively.

Finally, this work concludes that adding self and channel attention into the generator network, and utilizing MAE loss as adversarial loss does improve the quality of the image sequences being generated. Evenly distributing values according to the frame number to assign expression intensity value improves the consistency of image sequences being generated and also enables the generator to generate different frame size videos while remaining in the range [0,1].

Future Work

This work synthesizes facial expressions from a neutral face image but it can be expanded to generate facial expression from one peak expression to another peak expression if the peak expression is provided as input and the neutral image is the output then it can generate facial expression from one expression to another. Due to time constraints, the AI2VFEGAN model was only trained for four expressions; however, for it to be suitable for a wide range of real-world applications, it needs to be trained with additional expressions. Because of time constraints, the model is trained for 1000 epochs, although it can produce better results if trained for a longer period. The attention mechanisms help the network in producing good results. Channel attention has been demonstrated to be effective in retaining information-rich features in each layer but interprets each convolution layer as an independent operation, so integrating the holistic attention mechanism into the network might improve the result since it represents the holistic interdependencies across layers, channels, and positions, which comprises a (LAM) layer attention module and a (CSAM) channel-spatial attention module. Another important

aspect of this study is to synthesize facial expressions of non-frontal face photos; nevertheless, the local dataset was only created with frontal, near 45-degree, and 90-degree faces posed on both the left and right sides. In the future, the 15-, 30-, 60-, and 75-degree face postures should be considered.

SPECIAL ACKNOWLEDGMENT

This research work was funded by Adama Science and Technology University under grant number:

ASTU/SM-R/234/21

Adama, Ethiopia

REFERENCES

Agianpuye, S., & Minoi, J. L. (2013). 3D Facial expression synthesis: A survey. *2013 8th International Conference on Information Technology in Asia - Smart Devices Trend: Technologising Future Lifestyle, Proceedings of CITA 2013*. https://doi.org/10.1109/CITA.2013.6637552

Aifanti, N., Papachristou, C., & Delopoulos, A. (2010). Niki Aifanti , Christos Papachristou and Anastasios Delopoulos. *11th International Workshop on Image Analysis for Multimedia Interactive Services WIAMIS 10*, 1–4.

Averbuch-Elor, H., Cohen-Or, D., Kopf, J., & Cohen, M. F. (2017). Bringing portraits to life. *ACM Transactions on Graphics*, *36*(6). https://doi.org/10.1145/3130800.3130818

Baltruˇ, T. (n.d.). *OpenFace 2 . 0 : Facial Behavior Analysis Toolkit*.

Bao, J., Chen, D., Wen, F., Li, H., & Hua, G. (2018). Towards Open-Set Identity Preserving Face Synthesis. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 6713–6722. https://doi.org/10.1109/CVPR.2018.00702

Blanz, V., Basso, C., Poggio, T., & Vetter, T. (2003). Reanimating Faces in Images and Video. *Computer Graphics Forum*, *22*(3), 641–650. https://doi.org/10.1111/1467-8659.t01-1-00712

Bozorgtabar, B., Mahapatra, D., & Thiran, J. P. (2020). ExprADA: Adversarial domain adaptation for facial expression analysis. *Pattern Recognition*, *100*, 107111. https://doi.org/10.1016/j.patcog.2019.107111

Bulat, A., & Tzimiropoulos, G. (n.d.). *How far are we from solving the 2D & 3D Face Alignment problem ? (and a dataset of 230 , 000 3D facial landmarks)*. 1021–1030.

Cai, H., Bai, C., Tai, Y. W., & Tang, C. K. (2018). Deep Video Generation, Prediction and Completion of Human Action Sequences. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *11206 LNCS*, 374–390. https://doi.org/10.1007/978-3-030-01216-8_23

Chan, C., Ginosar, S., Zhou, T., & Efros, A. (2019). Everybody dance now. *Proceedings of the IEEE International Conference on Computer Vision*, *2019-Octob*, 5932–5941. https://doi.org/10.1109/ICCV.2019.00603

Chen, B., Wang, W., & Wang, J. (2017). Video imagination from a single image with transformation

generation. *Thematic Workshops 2017 - Proceedings of the Thematic Workshops of ACM Multimedia 2017, Co-Located with MM 2017, October*, 358–366. https://doi.org/10.1145/3126686.3126737

Chen, M., Li, C., Li, K., Zhang, H., & He, X. (2018). Double Encoder Conditional GAN for Facial Expression Synthesis. *Chinese Control Conference, CCC*, *2018-July*, 9286–9291. https://doi.org/10.23919/ChiCC.2018.8483579

Chen, Z. S., Hou, K. R., Zhu, M. Y., Xu, Y., & Zhu, Q. X. (2021). A virtual sample generation approach based on a modified conditional GAN and centroidal Voronoi tessellation sampling to cope with small sample size problems: Application to soft sensing for chemical process. *Applied Soft Computing*, *101*, 107070. https://doi.org/10.1016/j.asoc.2020.107070

Cheong, S. Y. (n.d.). *Hands-On Image Generation with TensorFlow*.

Choi, Y., Choi, M., Kim, M., Ha, J. W., Kim, S., & Choo, J. (2018). StarGAN: Unified Generative Adversarial Networks for Multi-domain Image-to-Image Translation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 8789–8797. https://doi.org/10.1109/CVPR.2018.00916

Choi, Y., Uh, Y., Yoo, J., & Ha, J. W. (2020). StarGAN v2: Diverse Image Synthesis for Multiple Domains. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 8185–8194. https://doi.org/10.1109/CVPR42600.2020.00821

Denton, E., & Birodkar, V. (2017). Unsupervised learning of disentangled representations from video. *Advances in Neural Information Processing Systems*, *2017-Decem*, 4415–4424.

Ding, H., Sricharan, K., & Chellappa, R. (2018). ExprGAN: Facial expression editing with controllable expression intensity. *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, 6781–6788.

Fan, L., Huang, W., Gan, C., Huang, J., & Gong, B. (2019). Controllable image-to-video translation: A case study on facial expression generation. *33rd AAAI Conference on Artificial Intelligence, AAAI 2019, 31st Innovative Applications of Artificial Intelligence Conference, IAAI 2019 and the 9th AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019*, 3510–3517. https://doi.org/10.1609/aaai.v33i01.33013510

Fu, C., Hu, Y., Wu, X., Wang, G., Zhang, Q., & He, R. (2021). High-Fidelity Face Manipulation with Extreme Poses and Expressions. *IEEE Transactions on Information Forensics and Security*, *16*, 2218–2231. https://doi.org/10.1109/TIFS.2021.3050065

Garrido, P., Valgaerts, L., Rehmsen, O., Thormählen, T., Pérez, P., & Theobalt, C. (2014). Automatic face reenactment. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 4217–4224. https://doi.org/10.1109/CVPR.2014.537

Gollapudi, S. (2019). Learn Computer Vision Using OpenCV. In *Learn Computer Vision Using OpenCV*. https://doi.org/10.1007/978-1-4842-4261-2

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2020). Generative adversarial networks. *Communications of the ACM*, *63*(11), 139–144. https://doi.org/10.1145/3422622

Hu, J., Shen, L., Albanie, S., Sun, G., & Wu, E. (2020). Squeeze-and-Excitation Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *42*(8), 2011–2023. https://doi.org/10.1109/TPAMI.2019.2913372

Huang, Q., Yang, D., Wu, P., Qu, H., Yi, J., & Metaxas, D. (2019). MRI reconstruction via cascaded channel-wise attention network. *Proceedings - International Symposium on Biomedical Imaging*, *2019-April*, 1622–1626. https://doi.org/10.1109/ISBI.2019.8759423

Isola, P., Zhu, J. Y., Zhou, T., & Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, *2017-Janua*, 5967–5976. https://doi.org/10.1109/CVPR.2017.632

Kalchbrenner, N., Van Den Oord, A., Simonyan, K., Danihelka, I., Vinyals, O., Graves, A., & Kavukcuoglu, K. (2017). Video pixel networks. *34th International Conference on Machine Learning, ICML 2017*, *4*, 2821–2829.

Kingma, D. P., & Ba, J. L. (2015). Adam: A method for stochastic optimization. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 1–15.

Krizhevsky Alex, Sutskever Ilya, & E. Hinton Geoffrey. (2012). *AlexNet*. https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

Lecun, Y., Bottou, L., Bengio, Y., & Ha, P. (1998). LeNet. *Proceedings of the IEEE*, *November*, 1–46.

Lee, A. X., Zhang, R., Ebert, F., Abbeel, P., Finn, C., & Levine, S. (2018). *Stochastic Adversarial Video Prediction*. http://arxiv.org/abs/1804.01523

Li, K., Xu, F., Wang, J., Dai, Q., & Liu, Y. (2012). A data-driven approach for facial expression synthesis in video. *Proceedings of the IEEE Computer Society Conference on Computer Vision*

and Pattern Recognition, 57–64. https://doi.org/10.1109/CVPR.2012.6247658

Li, M., Zuo, W., & Zhang, D. (2016). *Deep Identity-aware Transfer of Facial Attributes*. 1–14. http://arxiv.org/abs/1610.05586

Li, Y., Fang, C., Yang, J., Wang, Z., Lu, X., & Yang, M. H. (2018). Flow-grounded spatial-temporal video prediction from still images. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *11213 LNCS*, 609–625. https://doi.org/10.1007/978-3-030-01240-3_37

Liang, X., Lee, L., Dai, W., & Xing, E. P. (2017). Dual Motion GAN for Future-Flow Embedded Video Prediction. *Proceedings of the IEEE International Conference on Computer Vision*, *2017-Octob*, 1762–1770. https://doi.org/10.1109/ICCV.2017.194

Lotter, W., Kreiman, G., & Cox, D. (2017). Deep predictive coding networks for video prediction and unsupervised learning. *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, 1–18.

Lu, Z., Hu, T., Song, L., Zhang, Z., & He, R. (2018). Conditional expression synthesis with face parsing transformation. *MM 2018 - Proceedings of the 2018 ACM Multimedia Conference*, 1083–1091. https://doi.org/10.1145/3240508.3240647

Mathieu, M., Couprie, C., & LeCun, Y. (2016). Deep multi-scale video prediction beyond mean square error. *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, *2015*, 1–14.

Mirza, M., & Osindero, S. (2014). *Conditional Generative Adversarial Nets*. 1–7. http://arxiv.org/abs/1411.1784

Nandhini Abirami, R., Durai Raj Vincent, P. M., Srinivasan, K., Tariq, U., & Chang, C. Y. (2021). Deep CNN and Deep GAN in Computational Visual Perception-Driven Image Analysis. *Complexity*, *2021*. https://doi.org/10.1155/2021/5541134

Nirkin, Y., Keller, Y., & Hassner, T. (2019). FSGAN: Subject agnostic face swapping and reenactment. *Proceedings of the IEEE International Conference on Computer Vision*, *2019-Octob*, 7183–7192. https://doi.org/10.1109/ICCV.2019.00728

Olszewski, K., Li, Z., Yang, C., Zhou, Y., Yu, R., Huang, Z., Xiang, S., Saito, S., Kohli, P., & Li, H. (2017). Realistic Dynamic Facial Textures from a Single Image Using GANs. *Proceedings of the IEEE International Conference on Computer Vision*, *2017-Octob*, 5439–5448.

https://doi.org/10.1109/ICCV.2017.580

One, N., & Translation, M. I. (2019). *Unsupervised One-To-Many Image Translation*. 1–12.

OTBERDOUT, N., Daoudi, M., Kacem, A., Ballihi, L., & Berretti, S. (2020). Dynamic Facial Expression Generation on Hilbert Hypersphere with Conditional Wasserstein Generative Adversarial Nets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1–1. https://doi.org/10.1109/tpami.2020.3002500

Perarnau, G., van de Weijer, J., Raducanu, B., & Álvarez, J. M. (2016). *Invertible Conditional GANs for image editing*. *Figure 1*, 1–9. http://arxiv.org/abs/1611.06355

Pumarola, A., Agudo, A., Martinez, A. M., Sanfeliu, A., & Moreno-Noguer, F. (2018). GANimation: Anatomically-aware facial animation from a single image. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *11214 LNCS*, 835–851. https://doi.org/10.1007/978-3-030-01249-6_50

Qian, S., Lin, K. Y., Wu, W., Liu, Y., Wang, Q., Shen, F., Qian, C., & He, R. (2019). Make a face: Towards arbitrary high fidelity face manipulation. *Proceedings of the IEEE International Conference on Computer Vision*, *2019-Octob*, 10032–10041. https://doi.org/10.1109/ICCV.2019.01013

Qiao, F., Yao, N., Jiao, Z., Li, Z., Chen, H., & Wang, H. (2018). *Geometry-Contrastive GAN for Facial Expression Transfer*. http://arxiv.org/abs/1802.01822

Reda, F. A., Liu, G., Shih, K. J., Kirby, R., Barker, J., Tarjan, D., Tao, A., & Catanzaro, B. (2018). SDC-Net: Video prediction using spatially-displaced convolution. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *11211 LNCS*, 747–763. https://doi.org/10.1007/978-3-030-01234-2_44

Saito, M., Matsumoto, E., & Saito, S. (2017). Temporal Generative Adversarial Nets with Singular Value Clipping. *Proceedings of the IEEE International Conference on Computer Vision*, *2017-Octob*, 2849–2858. https://doi.org/10.1109/ICCV.2017.308

Shen, G., Huang, W., Gan, C., Tan, M., Huang, J., Zhu, W., & Gong, B. (2019). Facial image-to-video translation by a hidden affine transformation. *MM 2019 - Proceedings of the 27th ACM International Conference on Multimedia*, 2505–2513. https://doi.org/10.1145/3343031.3350981

Shen, W., & Liu, R. (2017). Learning residual images for face attribute manipulation. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, *2017-Janua*,

1225–1233. https://doi.org/10.1109/CVPR.2017.135

Siarohin, A., Lathuiliere, S., Tulyakov, S., Ricci, E., & Sebe, N. (2019). Animating arbitrary objects via deep motion transfer. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *2019-June*, 2372–2381. https://doi.org/10.1109/CVPR.2019.00248

Siarohin, A., Lathuilière, S., Tulyakov, S., Ricci, E., & Sebe, N. (2019). First order motion model for image animation. *Advances in Neural Information Processing Systems*, *32*(NeurIPS).

Song, L., Lu, Z., He, R., Sun, Z., & Tan, T. (2018). Geometry guided adversarial facial expression synthesis. *MM 2018 - Proceedings of the 2018 ACM Multimedia Conference*, 627–635. https://doi.org/10.1145/3240508.3240612

Tulyakov, S., Liu, M. Y., Yang, X., & Kautz, J. (2018). MoCoGAN: Decomposing Motion and Content for Video Generation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1526–1535. https://doi.org/10.1109/CVPR.2018.00165

Ververas, E., & Zafeiriou, S. (2020). SliderGAN: Synthesizing Expressive Face Images by Sliding 3D Blendshape Parameters. *International Journal of Computer Vision*, *128*(10–11), 2629–2650. https://doi.org/10.1007/s11263-020-01338-7

Villegas, R., Yang, J., Hong, S., Lin, X., & Lee, H. (2017). Decomposing motion and content for natural video sequence prediction. *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, 1–22.

Villegas, R., Yang, J., Zou, Y., Sohn, S., Lin, X., & Lee, H. (2017). Learning to generate long-term future via hierarchical prediction. *34th International Conference on Machine Learning, ICML 2017*, *7*, 5429–5449.

Walker, J., Marino, K., Gupta, A., & Hebert, M. (2017). The Pose Knows: Video Forecasting by Generating Pose Futures. *Proceedings of the IEEE International Conference on Computer Vision*, *2017-Octob*, 3352–3361. https://doi.org/10.1109/ICCV.2017.361

Wang, T. C., Liu, M. Y., Zhu, J. Y., Tao, A., Kautz, J., & Catanzaro, B. (2018). High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 8798–8807. https://doi.org/10.1109/CVPR.2018.00917

Wang, T., Liu, M., Zhu, J., Liu, G., Tao, A., Kautz, J., & Catanzaro, B. (n.d.). *Video-to-Video Synthesis*. 1–14.

Wang, X., Li, W., Mu, G., Huang, D., & Wang, Y. (2018). Facial expression synthesis by u-net conditional generative adversarial networks. *ICMR 2018 - Proceedings of the 2018 ACM International Conference on Multimedia Retrieval*, 283–290. https://doi.org/10.1145/3206025.3206068

Wang, Z., Bovik, A. C., Sheikh, H. R., & Simoncelli, E. P. (2004). Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, *13*(4), 600–612. https://doi.org/10.1109/TIP.2003.819861

Weng, W., & Zhu, X. (2021). INet: Convolutional Networks for Biomedical Image Segmentation. *IEEE Access*, *9*, 16591–16603. https://doi.org/10.1109/ACCESS.2021.3053408

Yan, Y., Huang, Y., Chen, S., Shen, C., & Wang, H. (2020). Joint Deep Learning of Facial Expression Synthesis and Recognition. *IEEE Transactions on Multimedia*, *22*(11), 2792–2807. https://doi.org/10.1109/TMM.2019.2962317

Yang, F., Bourdev, L., Shechtman, E., Wang, J., Metaxas, D., & Park, M. (n.d.). *Facial Expression Editing in Video Using a Temporally-Smooth Factorization*.

Yang, F., Metaxas, D., Wang, J., Shechtman, E., & Bourdev, L. (2011). Expression Flow for 3D-Aware Face Component Transfer. *ACM Transactions on Graphics*, *30*(4), 1–10. https://doi.org/10.1145/2010324.1964955

Yeh, R., Liu, Z., Goldman, D. B., & Agarwala, A. (2016). *Semantic Facial Expression Editing using Autoencoded Flow*. http://arxiv.org/abs/1611.09961

Yuan, Z., Jiang, M., Wang, Y., Wei, B., Li, Y., Wang, P., Menpes-Smith, W., Niu, Z., & Yang, G. (2020). SARA-GAN: Self-Attention and Relative Average Discriminator Based Generative Adversarial Networks for Fast Compressed Sensing MRI Reconstruction. *Frontiers in Neuroinformatics*, *14*(November). https://doi.org/10.3389/fninf.2020.611666

Zhang, H., Goodfellow, I., Metaxas, D., & Odena, A. (2019). Self-attention generative adversarial networks. *36th International Conference on Machine Learning, ICML 2019*, *2019-June*, 12744–12753.

Zhang, Yongqi. (2018). *XOGAN: One-to-Many Unsupervised Image-to-Image Translation*. http://arxiv.org/abs/1805.07277

Zhang, Yulun, Li, K., Li, K., Wang, L., Zhong, B., & Fu, Y. (2018). Image super-resolution using very deep residual channel attention networks. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *11211 LNCS*, 294–310. https://doi.org/10.1007/978-3-030-01234-2_18

Zhao, L., Peng, X., Tian, Y., Kapadia, M., & Metaxas, D. (2018). Learning to forecast and refine residual motion for image-to-video generation. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *11219 LNCS*, 403–419. https://doi.org/10.1007/978-3-030-01267-0_24

Zhao, Y. (2018). *A video prediction approach for animating single face image*.

Zheng, X., Guo, Y., Huang, H., Li, Y., & He, R. (2020). A Survey of Deep Facial Attribute Analysis. *International Journal of Computer Vision*, *128*(8–9), 2002–2034. https://doi.org/10.1007/s11263-020-01308-z

Zhou, Y., Song, Y., & Berg, T. L. (2018). Image2GIF: Generating Cinemagraphs Using Recurrent Deep Q-Networks. *Proceedings - 2018 IEEE Winter Conference on Applications of Computer Vision, WACV 2018*, *2018-Janua*, 170–178. https://doi.org/10.1109/WACV.2018.00025

Zhu, J., Park, T., Efros, A. A., Ai, B., & Berkeley, U. C. (n.d.). *Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks*.

Zhu, J. Y., Park, T., Isola, P., & Efros, A. A. (2017). Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks. *Proceedings of the IEEE International Conference on Computer Vision*, *2017-Octob*, 2242–2251. https://doi.org/10.1109/ICCV.2017.244

Zhu, J. Y., Zhang, R., Pathak, D., Darrell, T., Efros, A. A., Wang, O., & Shechtman, E. (2017). Toward multimodal image-to-image translation. *Advances in Neural Information Processing Systems*, *2017-Decem*(1), 466–477.

Appendix A: MUG facial expression database license agreement

MUG Facial Expression Database License Agreement

To advance the research efforts in facial expression recognition, the database will be made available to reseachers on a case-by-case basis. All requests for the database must be submitted in writing by the researchers institution, prior to the release of the database. To receive a copy of the database, the researcher (signatory) must agree to the conditions below, and sign the document. The database can be downloaded or received by mailing, after the receipt of the signed document.

Conditions:

1. **Redistribution:** The database will not be further distributed, published, copied or further disseminated in any way or form whatsoever, whether for profit or not. This includes further distributing, copying or disseminating to a different facility or organizational unit in the requesting university, organization or company.
2. **Requests and Access:** The signatory may only use the database after this License Agreement has been signed and returned to the MUG database administrator. All requests for the MUG database will be forwarded to the database administrator. When a group of researchers wants access to the database, each researcher must sign the License Agreement individually and will be given an individual username and password. The signatory may not grant anyone access to the database by giving out their username and password.
3. **Protection of identity:** The identity of the subjects in the database is strictly protected. Names are not released. The signatory agrees not to disclose the names of the persons in the database and to connect the data with the names, if I get this information through another source.
4. **Modification:** The signatory must not modify the database in any form.
5. **Commercial use:** The database is released for research purposes only. The signatory will not use the original data or modified copies for commercial purposes.
6. **Publication:** Publications include not only papers, but also presentations for conferences or educational purposes. The signatory may use screenshots of subjects in publications only if that particular subject has explicitly granted the permission to use his or her recordings in publications.
7. **Citations:** All documents and papers that report research on the MUG database must cite the following paper: N. Aifanti, C. Papachristou and A. Delopoulos, "The MUG Facial Expression Database," *in Proc. 11th Int. Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS)*, Desenzano, Italy, April 12-14 2010.

Printed Name: KIDIST ALEMAYEHU

Organization: ADAMA SCIENCE AND TECHNOLOGY UNIVERSITY

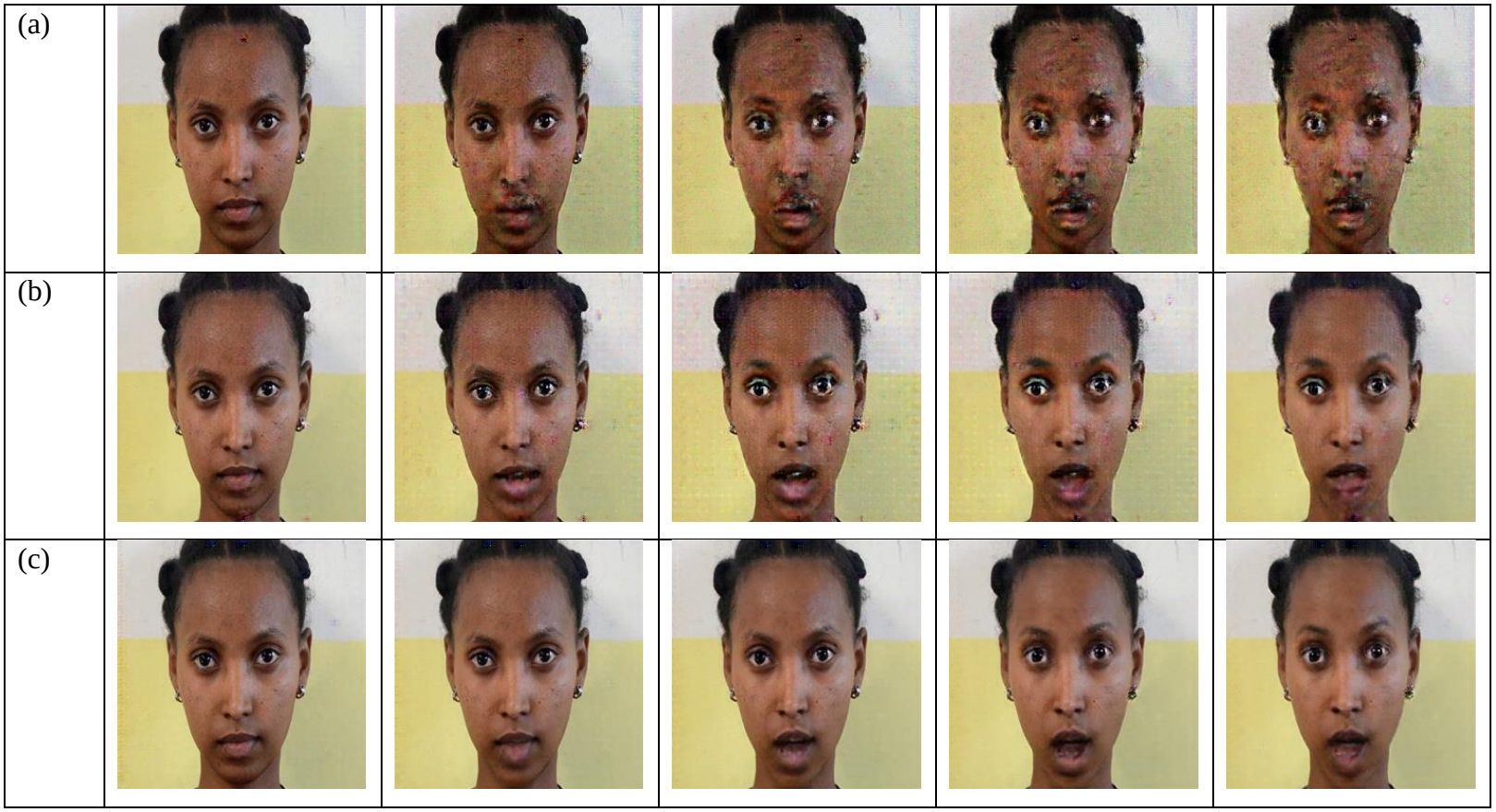
Address: ETHIOPIA

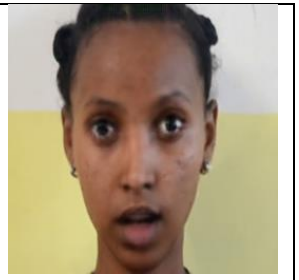
Email: kidistalemayehu@research.astu.edu.et

Signature: Date: 2/3/2021

Appendix B: Ablation Study

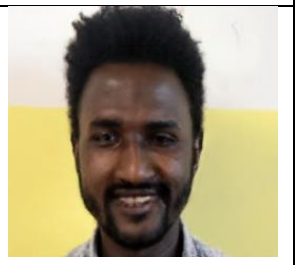
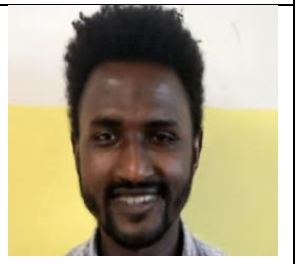
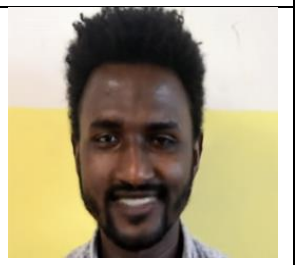
This ablation study has been made to come up to MAE loss to train both the generator and the discriminator network. Different loss functions MSE, relativistic average standard GAN, relativistic average LSGAN (Least square GAN), and smooth l1 loss were utilized to train the model. The MSE loss gave almost similar result to MAE loss visually but the discriminator loss remains zero after 300 epochs. The relativistic average standard GAN, relativistic average LSGAN, and the smooth l1 loss did not give good quality image sequences as the figure below shows.



(d)					
(e)					

(a) Shows the result obtained by relativistic average standard GAN, (b) shows the result obtained by relativistic average LSGAN, (c) shows the result obtained by smooth l1 loss, (d) shows the result obtained by Mean Square Error loss and (e) shows the result obtained by Mean Absolute Error loss

Appendix C: Result on different Epochs

100 epochs					
300 epochs					
600 epochs					
900 epochs					

The result on different epochs on the Local datasets, the label shows the corresponding epoch

Appendix D: Sample code

Train Generator Network

```
class UnetSkipConnectionBlock(nn.Module):

    def __init__(self, outer_nc, inner_nc, input_nc=None,

                 submodule=None, outermost=False, innermost=False,
norm_layer=nn.BatchNorm2d, gpu_ids=[]):

        super(UnetSkipConnectionBlock, self).__init__()

        self.device = torch.device('cuda:{}'.format(gpu_ids[0])) if
gpu_ids else torch.device('cpu')

        self.outermost = outermost

        self.innermost = innermost

        self.out_nc = outer_nc

        self.in_nc = inner_nc

        if type(norm_layer) == functools.partial:

            use_bias = norm_layer.func == nn.InstanceNorm2d

        else:

            use_bias = norm_layer == nn.InstanceNorm2d

        if input_nc is None:

            input_nc = outer_nc

        downconv_base = nn.Conv2d(input_nc, inner_nc, kernel_size=4,

                                  stride=2, padding=1, bias=use_bias)

        downrelu_base = nn.LeakyReLU(0.2, True)

        downnorm_base = norm_layer(inner_nc)

        downconv_res = nn.Conv2d(input_nc, inner_nc, kernel_size=4,

                                 stride=2, padding=1, bias=use_bias)

        downrelu_res = nn.LeakyReLU(0.2, True)

        downnorm_res = norm_layer(inner_nc)

        uprelu = nn.ReLU(True)

        upnorm = norm_layer(outer_nc)
```

```
        if outermost:
            upconv = nn.ConvTranspose2d(inner_nc * 2, outer_nc,
                                        kernel_size=4, stride=2,
                                        padding=1)
            down_base = [downconv_base]
            down_res = [downconv_res]
            up_all = [uprelu, upconv, nn.Tanh()]
        elif innermost:
            upconv = nn.ConvTranspose2d(inner_nc, outer_nc,
                                        kernel_size=4, stride=2,
                                        padding=1, bias=use_bias)
            down_base = [downrelu_base, downconv_base]
            down_res = [downrelu_res, downconv_res]
            self.attention = Self_Attn(512)
            up_all = [uprelu, upconv, upnorm]

        else:
            upconv = nn.ConvTranspose2d(inner_nc * 2, outer_nc,
                                        kernel_size=4, stride=2,
                                        padding=1, bias=use_bias)
            down_base = [downrelu_base, downconv_base, downnorm_base]
            down_res = [downrelu_res, downconv_res, downnorm_res]
            up = [uprelu, upconv, upnorm]
            up_all = up + [ChannelAttention(outer_nc, 16)]
        self.down_base = nn.Sequential(*down_base)
        self.down_res = nn.Sequential(*down_res)
        self.up_all = nn.Sequential(*up_all)
        self.sub = submodule
```

```
    def forward(self, x_base, x_res, base_stage, temporal_stage,
isTrain):
        if self.outermost:
            down_base = self.down_base(x_base)
            down_res = self.down_res(x_res)
            sub, feature = self.sub(down_base, down_res, base_stage,
temporal_stage, isTrain)
            sub_up = self.up_all(sub)
            return sub_up, feature
        if self.innermost:
            concat_1 = base_stage * x_base + temporal_stage * x_res
            if isTrain:
                shape = x_base.shape
                noise = torch.cuda.FloatTensor(shape) if
torch.cuda.is_available() else torch.FloatTensor(shape)
                torch.randn(shape, out=noise)
                concat_1 += noise * 0.01
            down_base = self.down_base(x_base)
            down_res = self.down_res(x_res)
            down = base_stage * down_base + temporal_stage * down_res
            if isTrain:
                shape = down_base.shape
                noise = torch.cuda.FloatTensor(shape) if
torch.cuda.is_available() else torch.FloatTensor(shape)
                torch.randn(shape, out=noise)
                down += noise * 0.01
            sub_up = self.up_all(down)
            return torch.cat([concat_1, sub_up], 1), down
        else:
            concat_1 = base_stage * x_base + temporal_stage * x_res
```

```
            if isTrain:
                shape = x_base.shape
                noise = torch.cuda.FloatTensor(shape) if
torch.cuda.is_available() else torch.FloatTensor(shape)
                torch.randn(shape, out=noise)
                concat_1 += noise * 0.01
            down_base = self.down_base(x_base)
            down_res = self.down_res(x_res)
            sub, feature = self.sub(down_base, down_res, base_stage,
temporal_stage, isTrain)
            sub_up = self.up_all(sub)
            return torch.cat([concat_1, sub_up], 1), feature
```

Self-attention implementation

```
def snconv2d(in_channels, out_channels, kernel_size, stride=1,
padding=0, dilation=1, groups=1, bias=True):
    return spectral_norm(nn.Conv2d(in_channels=in_channels,
out_channels=out_channels, kernel_size=kernel_size,
                                   stride=stride, padding=padding,
dilation=dilation, groups=groups, bias=bias))
def snlinear(in_features, out_features):
    return spectral_norm(nn.Linear(in_features=in_features,
out_features=out_features))
class Self_Attn(nn.Module):
    def __init__(self, in_channels):
        super(Self_Attn, self).__init__()
        self.in_channels = in_channels
        self.snconv1x1_theta = snconv2d(in_channels=in_channels,
out_channels=in_channels//8, kernel_size=1, stride=1, padding=0)
        self.snconv1x1_phi = snconv2d(in_channels=in_channels,
out_channels=in_channels//8, kernel_size=1, stride=1, padding=0)
```

```
        self.snconv1x1_g = snconv2d(in_channels=in_channels,
out_channels=in_channels//2, kernel_size=1, stride=1, padding=0)
        self.snconv1x1_attn = snconv2d(in_channels=in_channels//2,
out_channels=in_channels, kernel_size=1, stride=1, padding=0)
        self.maxpool = nn.MaxPool2d(2, stride=2, padding=0)
        self.softmax  = nn.Softmax(dim=-1)
        self.sigma = nn.Parameter(torch.zeros(1))
    def forward(self, x):
        _, ch, h, w = x.size()
        # Theta path
        theta = self.snconv1x1_theta(x)
        theta = theta.view(-1, ch//8, h*w)
        # Phi path
        phi = self.snconv1x1_phi(x)
        phi = self.maxpool(phi)
        phi = phi.view(-1, ch//8, h*w//4)
        # Attn map
        attn = torch.bmm(theta.permute(0, 2, 1), phi)
        attn = self.softmax(attn)
        # g path
        g = self.snconv1x1_g(x)
        g = self.maxpool(g)
        g = g.view(-1, ch//2, h*w//4)
        # Attn_g
        attn_g = torch.bmm(g, attn.permute(0, 2, 1))
        attn_g = attn_g.view(-1, ch//2, h, w)
        attn_g = self.snconv1x1_attn(attn_g)
        # Out
        out = x + self.sigma*attn_g
```

```
        return out
```

Channel attention implementation

```
class ChannelAttention(nn.Module):
    def __init__(self, num_features, reduction):
        super(ChannelAttention, self).__init__()
        self.module = nn.Sequential(
            nn.AdaptiveAvgPool2d(1),
            nn.Conv2d(num_features, num_features // reduction,
kernel_size=1),
            nn.ReLU(inplace=True),
            nn.Conv2d(num_features // reduction, num_features,
kernel_size=1),
            nn.Sigmoid())
    def forward(self, x):
        return x * self.module(x)
```