Disentangled Representation Based One-Shot Realistic Neural Talking Head Synthesis

Adugna Abe Belamo

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

September 2023

Adama, Ethiopia

Disentangled Representation Based One-Shot Realistic Neural Talking Head Synthesis

Adugna Abe Belamo

Advisor: Getinet Yilma (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

September 2023

Adama, Ethiopia

Declaration

I hereby declare that this Master Thesis entitled "**Disentangled Representation Based One-Shot Realistic Neural Talking Head Synthesis**" 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.

Name	Signature	Date

Recommendation

I, the advisor of this thesis, hereby certify that I have read the revised version of the thesis entitled **"Disentangled Representation Based One-Shot Realistic Neural Talking Head Synthesis"** prepared under my guidance by Adugna Abe 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.

_____	_____	_____
Major Advisor/Supervisor	Signature	Date

Approval Page

I, the advisor of the thesis entitled "**Disentangled Representation Based One-Shot Realistic Neural Talking Head Synthesis**" and developed by Adugna Abe Belamo, hereby certify that the recommendation and suggestions made by the board of examiners are appropriately incorporated into the final version of the thesis.

_____	_____	_____
Major Advisor/Supervisor	Signature	Date

We, the undersigned, members of the Board of Examiners of the thesis by Adugna Abe Belamo have read and evaluated the thesis entitled "**Disentangled Representation Based One-Shot Realistic Neural Talking Head Synthesis"** 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 and Engineering.

_____	_____	_____
Chairperson	Signature	Date
_____	_____	_____
Internal Examiner	Signature	Date
_____	_____	_____
External Examiner	Signature	Date

Finally, approval and acceptance of the thesis is 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

First and foremost, I would like to express my sincere gratitude to **God**, the **Almighty**, for providing me with courage, insight, and direction along this journey. I would also like to thank my advisor **Getinet Yilma (Ph.D),** for his invaluable mentorship, support, and encouragement for this study. He has always been available to answer my questions, provide helpful feedback, and offer valuable suggestions. I also want to thank him for guiding me in the right direction, providing timely feedback, and making insightful recommendations to help me meet the thesis requirements.

Then, I would like to thank **Prof. Yun Koo Chung,** for providing me with an excellent opportunity to work on this thesis and for providing continuous guidance throughout my studies. I extend my best wishes to him for all his future pursuits.

I also want to express my gratitude to my family, especially my parents, for their unwavering love, concern, and support as I was working on this thesis work. I am also grateful to my colleagues and friends, who have been there for me throughout this journey. They have shared their insights, feedback, and experiences with me.

Finally, I want to express my gratitude to everyone who took part in the study. Their efforts are priceless and much-appreciated contributions.

TABLE OF CONTENTS

ACKNOWLEDGMENT i

List of Tables vii

List of Figures viii

List of Sample Code Snippets ix

List of Acronyms and Abbreviations x

Abstract xii

CHAPTER ONE 1

1. INTRODUCTION 1

1.1. Background of the Study 1

1.2. Motivation of the Study 4

1.3. Statement of the Problem 5

1.4. Research Questions 6

1.5. Objective of the Study 6

1.5.1. General Objective 6

1.5.2. Specific Objectives 6

1.6. Scope and Limitations of the Study 7

1.6.1. Scope of the Study 7

1.6.2. Limitation of the Study 7

1.7. Application of Results 8

1.8. Organization of the Study 8

CHAPTER TWO 10

2. LITERATURE REVIEW AND RELATED WORKS 10

2.1. Introduction 10

2.2. Generative Adversarial Network (GAN) 11

2.2.1. Generative Adversarial Network Training 13
2.2.2. Conditional GAN (CGAN) 13
2.3. Image-to-Image Translation 14
2.4. Video Generation 15
2.5. Talking Head Synthesis 17
2.5.1. 2D-Based Talking Head Synthesis 17
2.5.2. 3D-Based Talking Head Synthesis 18
2.7. Related Works 19
2.8. Summary of Related Work 23
CHAPTER THREE 25
3. RESEARCH METHODOLOGY 25
3.1. Chapter Overview 25
3.2. Dataset Collection 25
3.3. Pre-processing Techniques 26
3.3.1. Resize Image 26
3.3.2. Normalization 26
3.3.3. Data Augmentation 27
3.4. Development Tools 27
3.4.1. Design Tools 27
3.4.2. Hardware Tools 28
3.4.3. Software Tools 28
3.5. Baseline Works 29
3.6. Feature Extraction Network 30
3.7.1. L1 31
3.7.2. Average Keypoint Distance (AKD) 31

3.7.3. Average Euclidean distance (AED) 32
CHAPTER FOUR 33
4. PROPOSED ARCHITECTURE 33
4.1. Chapter Overview 33
4.2. Proposed Model Architecture 33
4.2.1. Keypoint Detector 36
4.2.2. Background Motion Predictor 36
4.2.3. Dense Motion Network 37
4.2.4. Image Generation 40
4.2.5. Cross-Modal Attention Mechanism 42
4.2.6. Talking Head Synthesis via Disentanglement 44
4.2.7. Discriminator Network 45
4.3. Model Learning Functions 46
4.3.1. Perceptual Loss 47
4.3.2. Equivariance Loss 47
4.3.3. Background Loss 48
4.3.4. warp loss 48
4.3.5. Final loss 49
4.4. Training Pseudocode 49
CHAPTER FIVE 51
5. IMPLEMENTATION OF THE PROPOSED MODEL 51
5.1. Chapter Overview 51
5.2. Working Environments 51
5.3. Environment Setup 51
5.4. Training Procedure 52

5.5. Proposed Model Implementation 52
5.5.1. Keypoint Detector Implementation 54
5.5.2. Background Motion Predictor Implementation 55
5.5.3. Dense Motion Network Predictor Implementation 56
5.5.4. Image Generation Network Implementation 57
5.5.5. Cross-Modal Attention Implementation 58
5.5.6. Background Loss Implementation 59
5.5.7. Warp Loss Implementation 60
5.5.8. Discriminator Network Implementation 60
5.6. Hyperparameter Configuration 61
5.7. Experimental Class 62
CHAPTER SIX 64
6. RESULTS AND DISCUSSION 64
6.1. Chapter Overview 64
6.2. Result of the Study 64
6.2.1. Experimental Results 64
6.2.2. MRAA 64
6.2.3. MRAA+WL +BGL 65
6.2.4. MRAA+WL +BGL+CAM 66
6.2.5. MRAA+WL +BGL+CAM+ DR 67
6.3. Sample Training Log for MRAA+WL +BGL+CAM+ DR 69
6.4. Evaluation Metrics 71
6.4.1. L1 Error 71
6.4.2. Average Keypoint Distance 71
6.4.3. Average Euclidean Distance 72

6.5. Results Discussion 72
6.5.1. Discussion on L1 Error Results 72
6.5.2. Discussion on Average Keypoint Distance Results 73
6.5.3. Discussion on Average Euclidean Distance Results 74
6.5.4. Research Question Discussion 75
CHAPTER SEVEN 77
7. CONCLUSION AND FUTURE WORKS 77
7.1. Conclusion 77
7.2. Future Work 78
References 81
APPENDICES 87
Appendix A: Ablation Study 87
Appendix B: Result on different epoch 88
Appendix C: List of Sample Code 89

List of Tables

Table 2. 1: Summary of related works 23
Table 3. 1 Hardware Tools 28
Table 3. 2 Software Tools 28
Table 5. 1 Content of the generator architecture 53
Table 5. 2 Content of the discriminator architecture 54
Table 5. 3 Hyperparameter Configuration 62
Table 5. 4 List of experiment class 63
Table 6. 1: L1 error for all experiments 71
Table 6. 2: Average keypoint distance (AKD) for all experiments 71
Table 6. 3: Average Euclidean Distance (AED) for all experiments 72

List of Figures

Figure 1. 1: Example of realistic neural talking head using source image and driving video.....3
Figure 2. 1: Basic structure of GAN model.....12
Figure 2. 2: CGAN architecture 14
Figure 2. 3: Image-to-Image translation.....15
Figure 3. 1: Procedures of the overall research process workflow.....25
Figure 3. 2: VoxCeleb1 dataset samples 26
Figure 4. 1: Overall of the Proposed Solution Architecture.....35
Figure 4. 2: Keypoint Detector Architecture.....36
Figure 4. 3: Background Motion Predictor Architecture.....37
Figure 4. 4: Dense Motion Network Architecture.....39
Figure 4. 5: Generator Architecture.....41
Figure 4. 6: Cross-modal attention GAN 43
Figure 4. 7: Disentangled Representation Architecture 45
Figure 4. 8: Discriminator Architecture 46
Figure 6. 1: Results of MRAA using the VoxCeleb1 dataset for self-reenactment 65
Figure 6. 2: Results of MRAA using the VoxCeleb1 dataset for cross-identity 65
Figure 6. 3: Results of mraa+wl +bgl using the VoxCeleb1 dataset for self-reenactment.....66
Figure 6. 4: Results of mraa+wl +bgl using the VoxCeleb1 dataset for cross-identity.....66
Figure 6. 5: Results of mraa+wl +bgl+cam self-reenactment on the VoxCeleb1 dataset 66
Figure 6. 6: Results of mraa+wl +bgl+cam cross-identity on the VoxCeleb1 dataset.....67
Figure 6. 7: Results of mraa+wl +bgl+cam+dr for self-reenactment 68
Figure 6. 8: Results of mraa+wl +bgl+cam+ dr for cross-identity.....68
Figure 6. 9: Generator loss graph for MRAA+WL +BGL+CAM+ DR.....70
Figure 6. 10: Discriminator loss graph for MRAA+WL +BGL+CAM+ DR 70
Figure 6. 11: L1 Score of the for all experiments.....73
Figure 6. 12: AKD score of the for all experiments 74
Figure 6. 13: AED score of the for all experiments.....75

List of Sample Code Snippets

Sample code 5. 1 Keypoint Detector implementations 55
Sample code 5. 2 BG Motion Predictor implementations 56
Sample code 5. 3 Dense Motion Network Predictor implementations 57
Sample code 5. 4 Dense Motion Network Predictor implementations 58
Sample code 5. 5 Dense Motion Network Predictor implementations 59
Sample code 5. 6 Bg loss implementations 59
Sample code 5. 7 warp loss implementations 60
Sample code 5. 8 Discriminator network implementation 61

List of Acronyms and Abbreviations

AED	Average Euclidean Distance
AI	Artificial Intelligent
AKD	Average Keypoint Distance
ANN	Artificial Neural Network
AR	Augmented Reality
CAM	Cross-modal Attention Module
CG	Computer Graphics
CGAN	Conditional Generative Adversarial Network
CNN	Convolutional Neural Network
D	Driving Image
DL	Deep Learning
FOMM	First Order Motion Model for Image Animation
GAN	Generative Adversarial Network
GPU	Graphics Processing Unit
Lr	Learning Rate
LSTM	Long Short-Term Memory
ML	Machine Learning
MRAA	Motion Representations for Articulated Animation
PCA	Principal Component Analysis
RAM	Random Access Memory

ReLU	Rectified Linear Unit
ResNet	Residual Network
RGB	Red Green Blue
S	Source Image
SGD	Stochastic Gradient Descent
VAE	Variational Autoencoder
VGG	Visual Geometry Group
VR	Virtual Reality
2D	2-Dimension
3D	3-Dimension

Abstract

This study introduces a novel deep-learning model for realistic neural talking heads. Realistic neural talking heads synthesize a talking head video using the target person's appearance, which can be seen in the source image, and the output motion is controlled by a driving video. The primary focus of this study is to generate a video that preserves the original image's look by acquiring motion information from the driving video. Prior approaches in this area primarily depend on 2D representations, such as appearance and motion, acquired from the input image. Recent attempts have been made to execute motion transfer on any object utilizing unsupervised techniques without the use of prior information. However, the considerable difference in poses between the objects in the source and driving images remains a significant challenge for current unsupervised algorithms. Even the most recent method failed to achieve this correctly with good visual effects. In order to solve the problem of poor visual effects in the videos with the large-scale pose change, the GAN-based one-shot realistic neural talking heads model is proposed to mitigate these aforementioned issues. The proposed model employs cross-modal attention to preserve identity-related information and enhance the quality of the generated images. And also, use background and warp loss to reducing the background's noisy motion and encouraging the network to produce high-quality images. Additionally, to provide more precise and vivid visual effects, the multi-scale occlusion restoration module used in this study upsamples the low-resolution occlusion map to produce a multi-resolution occlusion map. Finally, in this study, disentangled representations were employed to facilitate animation and prevent the leakage of the driving object's appearance or shape. The experimental outcomes demonstrated that the proposed approach led to enhancements in several evaluation metrics, and the visual quality of the animated videos notably surpassed that of the MRAA. This model improves the MRAA baseline work from 0.040 to 0.034, from 1.28 to 1.13, and from 0.133 to 0.115, respectively, based on L1, AKD, and AED. Experiments demonstrate the superior performance of the proposed solution over the existing state-of-the-art methods on the Voxceleb1 benchmarked dataset with cross-modal attention, background and warp loss, multi-occlusion network, and disentangled representation.

Keywords: - Generative Adversarial Network, One-shot, MRAA, Disentangled Representation, 2D Representations.

CHAPTER ONE

1. INTRODUCTION

1.1. Background of the Study

A framework realistic talking head synthesis is a type of image animation, that aims to produce a synthetic human head video that contains the identity from a source image and pose and expression information from a driving video of the same object type, the generated video can mimic the motion in the driving video while simultaneously preserving the appearance or identity of the source image (Hong et al., 2022). It transfers the motion of the object in the driving video to the static object in the source image (Zhao & Zhang, 2022). In recent years, the computer vision community has paid much more attention to motion transfer.

Currently used image animation techniques can be classified into two groups: supervised methods and unsupervised approaches. More specifically, supervised approaches use a priori object knowledge, such as 3D models, landmarks, domain labels, and 3D optical flow, to focus on the animation of a particular object type, such as the human body, human face, and transfer motion. These techniques, which depend on annotated data, are only effective for specific objects such as faces and human bodies. Acquiring this annotated data or pre-trained keypoint extractors can be costly. Consequently, these methods are not applicable to objects that lack annotated data. Unsupervised motion transfer techniques, on the other hand, do not need to know the objects priori knowledge. By warping one image to rebuild another, these methods often learn intermediate motion representations keypoints and affine matrices between two images. (Tao et al., 2022).

This task has numerous potential applications, spanning areas such as augmented reality (AR), virtual reality (VR), telepresence, video conferencing, multiplayer gaming, image and video editing, as well as the production of animated movies. They can be further used for facial video compression and reconstruction in telecommunication (Meshry et al., 2021). Creating realistic talking head sequences is recognized as a challenging task for two main reasons. First the photometric, geometric, and kinematic complexity of human head is really high. This complexity results from modeling the mouth, hair, and clothing in addition to modeling faces,

for which there are numerous modeling approaches exist. The second complicating element is the human visual system's ability to detect even subtle discrepancies in the modeling of human appearance. The prevalence of non-photorealistic, cartoon-like avatars in many real-world teleconferencing systems can be attributed to the limited tolerance for modeling errors in human appearance (Zakharov et al., 2019).

Due to the advancements in generative adversarial networks (GANs) and their specific variant known as conditional GANs (cGANs) (Goodfellow et al., 2014), neural talking heads have emerged as a viable alternative to the conventional computer graphics approach. One-shot talking head generation is a technique that can create realistic and expressive facial animations from only one source face image. One reason why one-shot is recommended in talking head generation is that it can reduce the data and computational requirements for training and deploying the models. Unlike conventional methods that need a large amount of face images or videos to learn the facial features and motions, one-shot methods can leverage the existing knowledge from pre-trained models and adapt to new faces with minimal data. This can also enable the generation of talking heads for faces that are not available in the training data, such as celebrities, historical images, or fictional characters (J. Liu et al., 2023). In this study, we present a method for constructing talking head models using only a single frame, referred to as one-shot learning, accomplished within a limited training time.

For the past two decades, computer graphics methods have been employed to create talking head models by utilizing surface models based on meshes and texture maps. These resulting systems can be categorized into two distinct groups. After putting a lot of acquisition and design work into acquiring those particular persons, some are able to model them with an extremely high level of realism. Others have the ability to produce talking head models from just a few shots or a single image, but they do not strive for photorealism (Zakharov et al., 2020).

One-shot talking head generation is recommended for several reasons: data efficiency, practical use cases, resource conservation and also privacy and security. Recent 2D-based approaches, on the other hand, employ a subject-agnostic model that has the ability to animating unseen objects from a single image. Moreover, as these studies train an implicit model without the need for an explicit geometric representation, they have the capability to simulate the complete head, encompassing features such as hair, the interior of the mouth, and the inclusion of accessories

like glasses and earrings (Meshry et al., 2021). This study is a disentangled representation based on one-shot realistic neural talking head synthesis to produce highly realistic and customized talking head models of new persons, as well as portrait paintings using generative adversarial networks (GANs) techniques.

As illustrated in Figure 1, the motion observed in the driving videos in the top row is transferred to the source images in the second and third rows. In the animated videos found in the second and third rows, the objects from the source images follow the exact same motion seen in the driving videos. However, there are still some challenges associated with unsupervised techniques. One primary issue is that the motion representation lacks the necessary flexibility for the network to effectively learn the substantial pose difference between the objects in the source and driving images during the training process. Due to this flaw, there are significant differences between the feature domain of the driving image and the warped feature maps of the source image. Furthermore, as the surface area of the occlusion mask expands, the process of motion transfer starts to rely excessively on the network's ability to fill in missing details, leading to the second challenge: the network's limited inpainting capability.

Figure 1. 1: Example of realistic neural talking head using source image and driving video. Source: (Siarohin, Lathuilière, et al., 2019)

To address these issues and enhance performance, these research endeavors employ an attention module to enhance the creation of deformed feature maps, aiming to enhance image precision and attain a more consistent and resilient motion representation. Furthermore, we utilize a multi-occlusion network to repair image details across various scales, yielding more precise outcomes. This process involves adjusting the emphasis of feature maps to facilitate more efficient feature integration. These studies proposed to extend (Siarohin et al., 2021), training process involves

employing image reconstruction as a loss function and disentangling the motion information from the appearance information (Lorenz et al., 2019). This has opened up the possibility of animating a wider variety of object categories without the need for annotated data or domain knowledge, only videos depicting objects in motion during the training phase are needed. We also proposed the incorporation of supplementary loss functions to ensure that each module has a more distinct and defined role, thus aiding the network in generating high-quality images.

1.2. Motivation of the Study

Imagine a world where you can bring any idea, concept, or message to life with a virtual talking head that not only speaks but emotes, engages, and captivates your audience. This cutting-edge technology is poised to disrupt industries, transform education, enhance entertainment, and revolutionize communication. One possible motivation for talking head generation is to enable people to communicate more effectively and naturally across different languages, cultures, and distances. Imagine being able to speak to anyone in the world with your own voice and face, but in their native language and with their preferred style. You could also create personalized avatars for yourself or others, and use them to express your emotions, opinions, and stories in various scenarios. Talking head generation could also enhance the accessibility and diversity of online content, by allowing people to create and consume videos in different languages and with different faces. Recent work has demonstrated that by training generative convolutional neural networks across large datasets of images, they are able to produce numerous realistic human head images, but in practical scenarios, this has to be learned from a one images of a person.

There are algorithms that address these issues, but they require constructing head sequences with varying amounts of motion, head rotation, etc. and are still hard to achieve. Up until recently, the animation methods necessary to produce such effects needed a skilled professional, specialized equipment, software, and a lot of work. (Siarohin et al., 2021). Recent work has focused on using unsupervised motion transfer to avoid the necessity for ground truth data. There is no need for prior knowledge about objects, which typically includes landmarks, semantic segmentations, and parametric 3D models. Even though an abundance of research is presented to resolve these practical issues, this issue remains one of the most challenging when real-life scenarios are considered. This study tries to solve this problem by extending the solutions presented in previous works explicitly to talking head models.

1.3. Statement of the Problem

Personalized photorealistic talking head models or head reenactment are a hot or active research area as well as a difficult research problem. Even though recent research on neural talking heads has yielded encouraging results, they sometimes produce visuals that do not preserve the subject's identity in the source photos (Meshry et al., 2021). Although there are techniques for overcoming such difficulties, it is still difficult to create head sequences with sufficient motion, head rotation, etc. CNN need to be trained on enormous networks with tens of millions of parameters for both the generator and discriminator in order to perform such tasks, which means they require several minutes of video or a large dataset of photographs and a GPU to train. The main goal is not only to generate a photorealistic talking head; the model must be resistant to occlusions, changes in position, and adverse lighting conditions. Also, it should preserve the identity of the target person as much as possible. Still, their method suffers from a lack of details in the generated output and an identity mismatch (Rehusevych & Kupyn, 2020). The majority of existing approaches, however, are target-specific and incapable of producing photographs of previously unseen individuals. They also have faults, including blurriness and misaligned facial details.

Recent efforts have been made to achieve motion transfer on various objects through unsupervised techniques, without relying on prior information. However, a significant challenge persists in these unsupervised methods due to the considerable large pose change between the objects depicted in the source and driving images (Zhao & Zhang, 2022). However, there are still certain difficulties associated with unsupervised methods. First, there is a limited adaptability in the way motion is represented, making it difficult for the network to effectively understand the substantial pose differences between the objects featured in the source and driving images during the training process. This limitation results in significant difference between the feature domain of the driving image and the warped feature maps, posing a challenge. Additionally, as the occlusion mask's surface area grows, motion transfer becomes overly dependent on the network's ability to do inpainting, which brings us to the second issue: insufficient inpainting capabilities. Furthermore, current methods operate under the assumption of static backgrounds, meaning there is no camera movement involved. This assumption results in background motion information leakage into one or more of the detected keypoints. Finally,

absolute motion transfer reduces the accuracy of the source identity by incorporating the identity of the driving object into the resulting sequence.

This thesis aims to address the animation problem using disentangled representation based one-shot realistic neural talking head synthesis by offering novel solution focus on reducing the identity mismatch gap, temporal continuity and improving the photorealism of the synthesized faces.

1.4. Research Questions

This study addresses the problem by answering the following question:

RQ1: What is the effect of cross-modal attention mechanism in the realistic neural talking head synthesis using GAN model?

RQ2: What could be the effect of adding warp and background loss to improve the quality of the generated image?

RQ3: Can applying disentangled representation improves the quality of realistic neural talking head synthesis?

1.5. Objective of the Study

1.5.1. General Objective

The general objective of the study is to improve a GAN-based model for personalized photorealistic neural talking head models from a one-shot image using disentangled representation.

1.5.2. Specific Objectives

The subsequent goals are specific objectives aimed at accomplishing the general objective.

- To identify effective learning constraint to improve the quality of identity preservation, robustness to pose variations of neural talking head.
- To embed an attention mechanism, that can provide effective learning to the GAN model for achieving better preservation of facial structures.
- To design and implement a GAN based approach for neural talking head with new losses and unsupervised manner.

- To design GAN based model by adopting disentangled representation to enhance the animation quality.
- To evaluate the performance of the baseline work and the proposed GAN based model using different evaluation metrics.

1.6. Scope and Limitations of the Study

1.6.1. Scope of the Study

The goal of this study is to develop customized photorealistic neural talking head models using one-shot images of a source subject and a driving sequence that may be derived from another subject. The source subject will be represented in a photorealistic video under the poses and expressions of the driving sequence, or by systems that can produce convincing video sequences that mimic a specific person's speech and facial expressions. It animates the static object in the source image based on the motion in the driving video.

This study has four key performance advantages over existing methods. First, it can achieve better subject-agnostic model performance, meaning that it can generate high-quality talking head images for any face without fine-tuning. Second, it can achieve better fine-tuned performance with less data, meaning that it can further improve the quality of the talking head images by using only a one target face images for adaptation. Third, it can demonstrate robustness to pose variations, meaning that it can handle different head poses and orientations without losing the facial details and expressions. Fourth, it can ensure improved identity preservation, meaning that it can retain the identity and appearance of the source face image in the generated talking head.

1.6.2. Limitation of the Study

The limitation of this study didn’t consider audio, it may not be able to capture the full range of facial expressions and emotions that a human speaker would produce. Audio is a one-dimensional signal that mainly conveys the acoustic features of speech, such as pitch, intensity and duration. However, facial expressions and emotions are multi-dimensional and multimodal phenomena that depend on various factors, such as context, personality, culture, and intention. Therefore, it may be difficult to infer the appropriate facial expression and emotion from audio alone, without additional information or cues.

1.7. Application of Results

The proposed model is expected to find a wide range of applications in the following fields:

- **Telepresence Technology:** Telepresence enables you to remain in a single location while having the ability to perceive and behave as if you were physically present in another location. This represents the highest degree of video communication technology, offering significantly enhanced audio and visual quality compared to what you'd experience in conventional video conferencing.
- **AR/VR:** In an ever more digitized world, the terms Augmented Reality and Virtual Reality hold significant importance. Augmented reality (AR) involves the fusion of real-world and virtual elements, and it is a technology that functions effectively on both desktop computers and mobile devices. A new reality is simulated using virtual reality (VR). The user can see and engage in the digital environment by using a VR screen.
- **Video Conferencing:** Allows for the transmission of text and static images between two places. It has developed into a crucial platform for online social networking and distant teamwork. By transmitting just, the essential keypoints and then reconstructing the original video at the receiver's end, it can achieve a reduction in bandwidth usage compared to the industry standard, all while maintaining visual quality.
- **Animated Movie Production:** Creating a talking head animation of an anime character using only one or a few reference images, the goal is to mimic the dynamic head movements and facial expressions commonly seen in anime characters.
- **Puppeteering:** Generating a video representation of an individual by utilizing the facial landmark movements of another person (Zakharov et al., 2019). puppeteering of photographs and portrait paintings to bring still photographs to life and it can be used to regenerate memories of someone who is dead.
- **Entertainment:** From video games to animated films, entertainment will never be the same. Imagine characters that interact with you, respond to your emotions, and make the experience truly immersive. Consequently, these prospects have garnered significant attention from the academic community as well.

1.8. Organization of the Study

This study is structured into seven chapters, each serving a distinct purpose:

Chapter One: In this initial chapter, we provide an overview of the study and explain the reasons behind our chosen approach. We outline what the study aims to achieve and why.

Chapter Two: This chapter delves into the background literature and related research on topics such as image-to-image translation, the generation of age-progressed face images, the theoretical foundations of deep learning, and the use of Generative Adversarial Networks (GANs).

Chapter Three: Here, we provide a comprehensive description of our research methodology. This includes details on how we collected our dataset and the metrics we used to evaluate our model's performance.

Chapter Four: In this section, we provide a detailed description of the model we propose. We discuss its architecture, the loss functions it employs, and provide pseudocode for its training process.

Chapter Five: This chapter focuses on the practical implementation of our proposed solution. We explain how we set up the experimental environment for the study and provide code snippets that illustrate the implementation of our solution.

Chapter Six: Here, we present and analyze the results obtained from our proposed solution. We also compare these results with previous research to provide context and insights.

Chapter Seven: Finally, in this concluding chapter, we summarize the study's findings and draw conclusions. We also explore potential future avenues for research based on the outcomes of our work.

CHAPTER TWO

2. LITERATURE REVIEW AND RELATED WORKS

2.1. Introduction

This chapter delves into a comprehensive exploration of the field of human neural talking head reenactment. It aims to provide a thorough comprehension of the subject by examining current methods, challenges, outcomes, limitations, identified deficiencies, prominent application domains, and relevant prior research. The primary goal of this chapter is to establish an in-depth grasp of the research area, pinpoint its notable strengths and weaknesses, and propose a solution to address the existing gaps in the system. While the review may touch upon various theories and concepts, its central focus revolves around the examination of existing systems, the analysis of problems or obstacles, and the identification of areas where improvements are needed.

In the initial stages of software development, programs were constructed with rigidly programmed commands and rules expressed through mathematical formulas. These approaches were limited in their ability to replicate complex real-world scenarios and data convincingly. To address this limitation, a new paradigm in software development emerged, posing the question of whether machines could comprehend information in a manner akin to humans, a concept known as Artificial Intelligence or AI. AI has witnessed significant advancements in recent years, enabling AI applications to excel in tasks like data collection and extraction, as well as acquiring knowledge through a process referred to as machine learning (VanderPlas, 2016).

Machine learning (ML) is an expansive research field with a wide array of learning capabilities, and it continues to evolve. Among the various learning abilities within machine learning, unsupervised learning stands out. This type of learning involves the task of organizing unstructured data by clustering it based on inherent patterns or information it contains. Another facet of machine learning is supervised learning, which differs from unsupervised learning in that each data input is associated with a corresponding output label (represented as x, y). In supervised learning, the network's objective is to learn how to map from input x to its respective label y during testing, typically utilizing a paired dataset.

Semi-supervised learning represents another approach in machine learning where a portion of the data is labeled, while the remaining portion is left unlabeled. Despite the significant progress made in the field of machine learning, it still faces challenges when it comes to handling intricate data types such as images and videos. To tackle these complex data forms, deep learning has emerged as an alternative subfield within machine learning.

Deep learning represents a distinct approach to learning from data by employing sequential filter layers, as opposed to previous methods that focused on learning from data representations. In deep learning, these filters progressively capture information from data, creating a hierarchical hierarchy of features. These features are crucial for extracting high-level information. The inception of deep learning can be traced back to artificial neural networks (ANNs). Convolutional Neural Networks (CNNs), first introduced by Yann LeCun (Bengio & Haffner, 1998) in 1989 for recognizing handwritten digits, initially faced limitations due to the scarcity of large datasets and limited computing capabilities during that era. However, CNNs gained widespread popularity after the breakthrough achieved by AlexNet (Krizhevsky et al., 2012) in 2012 when it won the ImageNet competition using CNN-based image classification. AlexNet's success shed light on the immense potential of deep learning, inspiring numerous scientists and researchers. Today, we have advanced models and networks designed for tasks like face detection, object classification, and recognition.

The most noteworthy advancements in deep learning methods are generative models. These models are designed with the objective of understanding and learning the underlying distribution of data in a training set, allowing them to generate new data points that exhibit some level of variability. One prominent example of a deep generative model is the Generative Adversarial Network (GAN), introduced by Goodfellow (Goodfellow et al., 2014) and his colleagues in 2014. GAN operates on a game theory concept, aiming to discover the Nash equilibrium between two neural networks: the Generator (G) and the Discriminator (D).

2.2. Generative Adversarial Network (GAN)

In 2014, Ian Goodfellow (Goodfellow et al., 2014) introduced the concept of Generative Adversarial Networks (GANs). GANs consist of two neural networks, the Generator (G) and the Discriminator (D), which work together in a collaborative training process resembling a zero-sum game. In this setup, the Generator's objective is to generate synthetic data that closely

resembles real data, while the Discriminator's goal is to differentiate between genuine and generated data.

A discriminator network operates as a standard classifier, aiming to differentiate between real and generator-produced data. During training, it uses actual data as positive examples and generated data as negative examples. There are two associated loss functions with the discriminator: the generator loss and the discriminator loss. In the discriminator's training phase, it focuses solely on minimizing the discriminator loss while ignoring the generator loss. If the discriminator wrongly categorizes generated data as real, it incurs a penalty through the discriminator loss, and the reverse holds true. The discriminator then adjusts its weights through backpropagation to improve its ability to distinguish real samples from generated ones. On the other hand, the generator network is trained to create synthetic samples that closely resemble real data, starting from a random vector. Its primary goal is to deceive the discriminator into classifying its output as genuine. The generator loss penalizes the generator if it fails to trick the discriminator into believing its samples are real.

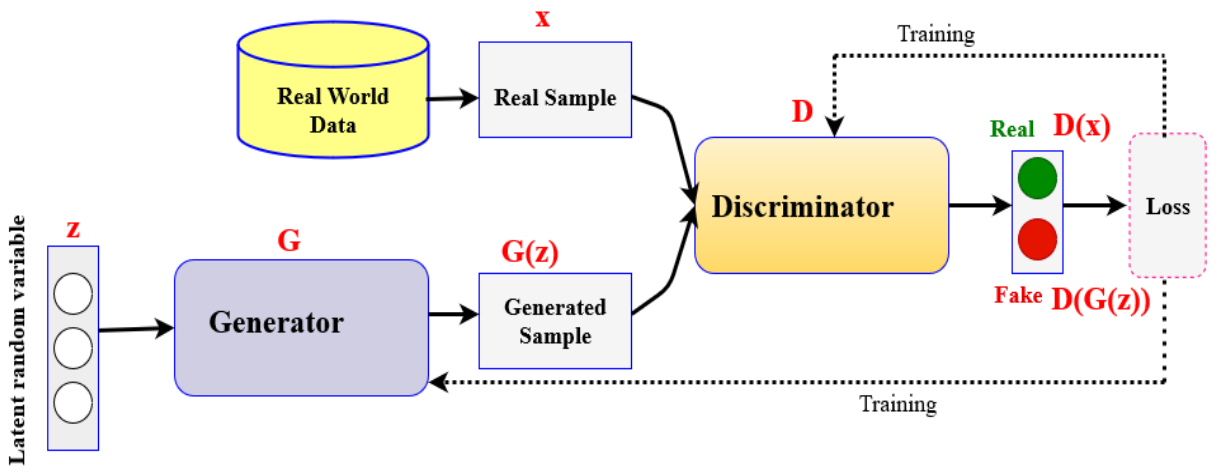


Figure 2. 1: Basic structure of GAN model
(Source: adapted from ((Nandhini Abirmi et al., 2021)))

The loss function of the GAN training:

$$\min_{G}\max_{D} V(G,D) = E_x[\log(D(x))] + E_z\left[\log\left(1 - D\left(G(z)\right)\right)\right] \qquad \text{eq. (2.1)}$$

2.2.1. Generative Adversarial Network Training

The training of Generative Adversarial Networks (GANs) consists of two distinct phases. In the first phase, the generator is kept static, and the training focuses solely on the discriminator. This means that generator training is temporarily halted, and only forward passes are performed without any backward propagation. In the second phase, the roles are reversed: the discriminator is frozen, and the generator is trained to produce more realistic outputs that can deceive the discriminator. These training steps are part of an iterative process, and they repeat to refine the GAN's performance.

1) Train the Discriminator:

a. Select a random mini-batch of real instances: x.

b. Create a mini-batch of fake instances by taking a mini-batch of random noise vectors, denoted as z. (z) = $xfake$.

c. Calculate the classification losses for both the real instances (x) and the generated instances $D(xfake)$, and then propagate the overall error backward to adjust the parameters $\theta^{(D)}$ in order to minimize the classification loss.

2) Train the Generator:

a. Create a set of fake instances in a mini-batch by using random noise vectors, typically denoted as z. (z) = $xfake$.

b. Calculate the classification loss for the generated instances $xfake$, and then propagate this loss backward to update the parameters $\theta^{(G)}$ with the goal of maximizing the classification loss.

2.2.2. Conditional GAN (CGAN)

A Conditional Generative Adversarial Network (Conditional GAN) extends the traditional GAN framework by incorporating extra information, denoted as y, into both the generator and discriminator. This additional information y, can take various forms, such as class labels in generating images of people (e.g., distinguishing between male and female) or digits when generating handwritten digit images. To implement this conditioning y, is provided as an additional input layer to both the generator and discriminator. The goal is for the generative model to learn how to generate new samples within a specific domain while taking into account this supplementary information y, as introduced in the work by (Mirza & Osindero, 2014) in 2014.

Within the generator, the input noise pz(z) and the additional information y are merged together into a shared hidden representation. The adversarial training process offers significant flexibility in determining how this hidden representation is constructed. On the other hand, in the discriminator, both the data x and the supplementary information y are used as inputs and are fed into a discriminative function for evaluation.

A two-player minimax game's primary objective would be to:

$$\min_{G}\max_{D} v(D,G) = E_{x\sim pdata(x)[log\, D(x|y)]+E_{z\sim p_{(z)}}[log(1-D(G(z|y)))]} \quad \text{eq. (2.2)}$$

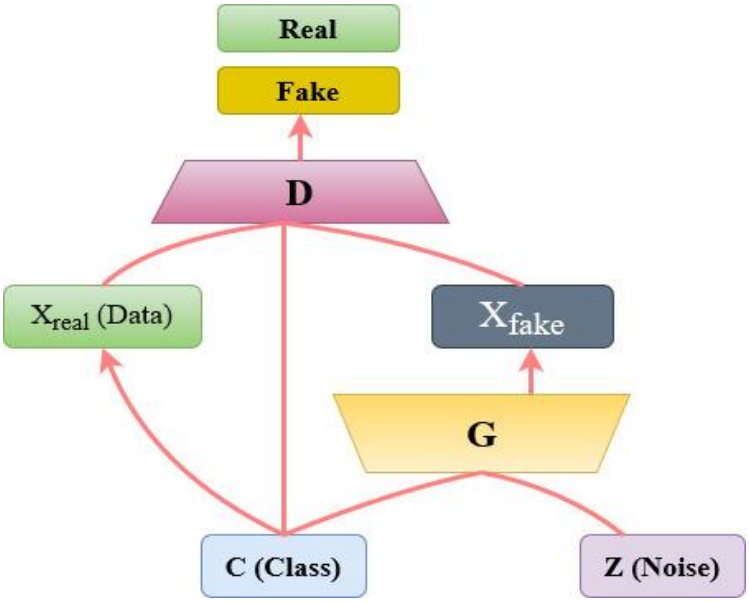


Figure 2. 2: CGAN architecture
(Source: adapted from (Mirza & Osindero, 2014))

GAN models can also be conditioned based on input content, which enables a wide range of applications, including tasks like translating images into other images, generating videos, and transforming images into videos.

2.3. Image-to-Image Translation

The process of converting images from one domain to another, like transforming label maps into photographs, sketches into images of shoes, or summer scenes into winter landscapes, is referred to as image-to-image translation.

Pix2pix (Isola et al., 2017), developed by Isola et al., introduced a novel approach to image-to-image translation by incorporating Conditional Generative Adversarial Networks (CGANs). This model utilized a patch-wise discriminator (often referred to as PatchGAN), which aimed to discern details within nearby image patches rather than evaluating the entire image as a whole. This strategy significantly reduced the computational load on the discriminator, as analyzing local patches demanded considerably less model capacity compared to evaluating the entire image. Building upon Pix2pix, a subsequent model known as Pix2pixHD (T. C. Wang et al., 2018), further improved image translation quality. Pix2pixHD achieved this by implementing a multi-scale discriminator and a generator that operated in a coarse-to-fine manner, ultimately resulting in more faithful and high-quality image translations.

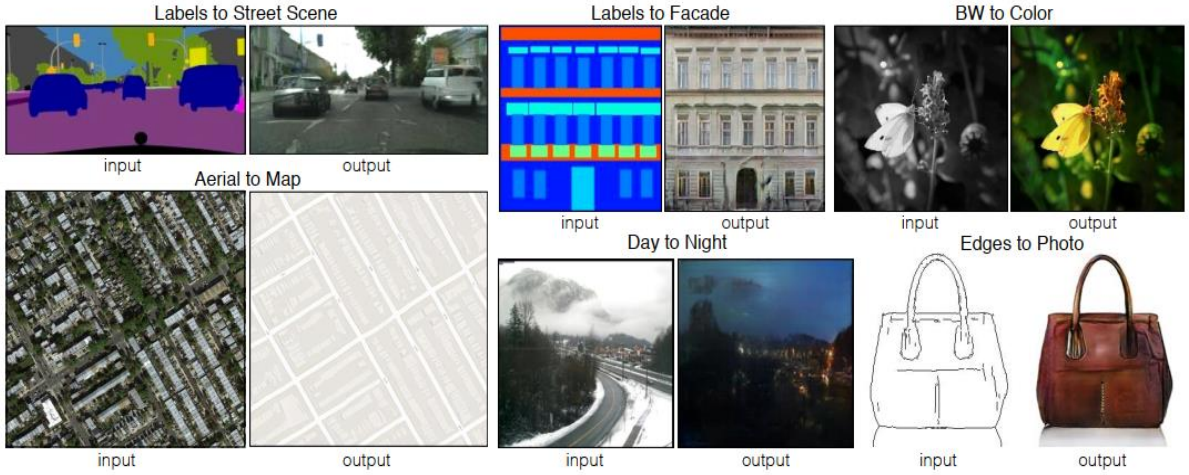


Figure 2. 3: Image-to-Image translation
(Source: adapted from (Isola et al., 2017))

2.4. Video Generation

There are three main categories here: 1) unconditional video synthesis models, which turn random noise samples into video clips; 2) future video prediction models, which create future video frames based on the ones that have already been seen; and 3) vid2vid models, which turn semantic input videos into photorealistic videos (T. Wang et al., 2019).

Unconditional video synthesis, as proposed by (Saito et al., 2017), relies on the utilization of random noise to create sequences. This approach employs two separate generators: an image generator and a temporal generator. The temporal generator takes a single latent vector as input and generates a set of latent vectors, each of which corresponds to individual frames within a

video sequence. These generated latent vectors are then processed by the image generator to produce the final video. However, since this approach is required to capture and model all the spatial and temporal information present in a video, the resulting videos tend to be relatively short or exhibit limited and constrained motion patterns.

The concept of future frame prediction is closely linked to the problem of generating videos, where the goal is to anticipate upcoming sequences of frames based on information from past frames. In the context of video sequences, these methods aim to create a series of images that form a coherent continuation of the input video. In earlier approaches (Srivastava et al., 2015); (Oh et al., 2015); (Kalchbrenner et al., 2017), the focus was on directly predicting the pixel values in the future frames. However, a more recent development by Villegas et al. (Villegas et al., 2017) introduced a hierarchical video prediction model that involves two stages. First, it predicts the motion of a set of landmarks using a Long Short-Term Memory (LSTM) model, and then it generates images based on these predicted landmarks.

Video-to-video models represent a logical extension of image-to-image translation tasks. In the context of video translation, the objective is to understand the appearance of objects within a scene and the realistic motion that occurs between consecutive frames. A straightforward approach to video-to-video translation involves applying image-to-image translation techniques independently to each frame of the input videos, without explicitly modeling relationships between these frames. More recently, there have been efforts to address the challenge of conditional video generation. For instance, (W. Wang et al., 2018) a combination of a recurrent neural network (RNN) and a Variational Autoencoder (VAE) was employed to generate facial videos. Another approach known as MocoGAN (Tulyakov et al., 2018), decomposes the video sequence into separate components for motion and content and generates motion using a fixed latent space along with a sequence of latent spaces, allowing for more nuanced control over video generation.

Since we also want to create video sequences using a deep learning architecture, our approach is closely similar to these earlier research work. The task we take on, however, is more difficult since talking head synthesis demands for decoupling, modeling and recombining motion and content information.

2.5. Talking Head Synthesis

More precisely, talking head synthesis is the process of automatically creating videos by merging motion patterns from a driving video with appearances taken from source images. For instance, a person's face image can be animated to mimic the gestures and poses of another person (Siarohin, Lathuilière, et al., 2019). The procedure generally consists of three basic steps, a modified source face is produced by: 1) producing a representation of the source face identity, 2) extracting and encoding the driving face's motion, and 3) combining the identity and motion representations. The output quality is significantly influenced by each component (Tripathy et al., 2021).

Due to its capacity to produce photorealistic images, generative adversarial networks have found success in this area. They have succeeded in generating unconditional faces with great quality and resolution (Hao et al., 2020). To solve this issue, few-shot talking head synthesis techniques have been created in the recent work (Wiles et al., 2018); (Zakharov et al., 2019). Wiles *et al*. (Wiles et al., 2018) propose a model, namely X2Face, it has the ability to direct the input source image to a target expression using facial landmarks or audio. Their model initially learns the frontalization of the source identity rather than immediately learning the transformation of expressions. Synthesizing frontal facing images of faces in a single unconstrained image is known as frontalization. Once the target expression is known, it creates an intermediate interpolation map that will be used to transfer the frontalized face. Zakharov *et al*. (Zakharov et al., 2019) provide a few-shot learning method that enables face renactment from a few number or even just one of source images.

The existing methods for realistic talking head synthesis can be divided into 2D and 3D-based methods.

2.5.1. 2D-Based Talking Head Synthesis

These methods work with an implicit head model and don't need a proxy geometry. As a result, they can synthesize the entire head, including dynamic areas like the interior mouth, eyes and hair. Additionally, they can model other accessories that can be wearable, such as earrings, hats, and eye glasses (Meshry et al., 2021). More recent works (Zakharov et al., 2019); (Siarohin, Lathuilière, et al., 2019); (Burkov et al., 2020) learn subject-agnostic models that can animate unseen objects using only a single or a few number of images. When compared to the 3D-based

subject-specific models, these techniques lack in terms of quality and identity preservation. According to the synthesis method, 2D-based techniques can be divided into warping-based (Siarohin, Lathuilière, et al., 2019); (Ha et al., 2020); (Wiles et al., 2018) and direct synthesis (Zakharov et al., 2019); (Burkov et al., 2020).

Direct synthesis approach generates an image employing a sequence of convolutional operators, interspersed with elementwise non-linearities, and normalizations. Injecting person identity information into such an architecture can be done either laboriously (in the many-shot scenario) or by employing adaptive normalizations conditioned learning on person embeddings. The majority of face reenactment methods use warping-based methods, which are another common generator architecture type. An input image or a recovered canonical posture is warped via warping-based approaches to create new poses. Although techniques that use warping can create talking head sequences from a single image, they are only capable of handling a certain degree of motion, head rotation, and occlusion without producing observable abnormalities.

2.5.2. 3D-Based Talking Head Synthesis

Such methods (Meshry et al., 2021) animate a target subject using 3D geometric representations as a proxy. Common geometric representations, including 3D morphable models (3DMM)(Vetter, 1999) simply model the area of the face and omit difficult regions like the interior of the mouth, eyes, and hair. The task of obtaining a precise geometry of these locations is expensive and difficult. As a result, such approaches are either unable to synthesize or have poor performance in such regions.

2.6. Feature Disentanglement Representation

The objective of this research topic is to separate the representation of a human face into two distinct components: one related to identity (content) and the other related to pose or expression (attribute). In the context of DR-GAN (Tran et al., 2017), when provided with information about the pose view and the corresponding human face image, an encoder-decoder network is employed to acquire a representation of identity that is invariant to pose variations. This type of representation is particularly valuable for face recognition tasks that involve significant differences in pose. Apart from applications like head or face reenactment, there exists a substantial body of research focused on learning disentangled representations. Some notable

works in this area involve the learning of latent descriptors for pose or shape for various object classes using video datasets. Certain approaches, as exemplified by (M. Y. Liu et al., 2019), strive to achieve disentanglement of content and style (which can roughly correspond to the separation of shape and texture) through the use of adversarial and cycle-consistency losses, following the principles introduced by (Goodfellow et al., 2014) and further developed by (Huang et al., 2018) and (Zhu et al., 2017).

2.7. Related Works

The typical computer graphics (CG) pipeline has been replaced in recent years by a new method called neural talking heads, which aims to generate visuals with high realism while using the least amount of input data possible. As talking heads techniques developed over time, they became less and less dependent on input data. Although one-shot neural talking head systems currently in use produce impressive results, there are still several difficult issues to solve.
The X2Face approach (Wiles et al., 2018) applies warping twice, first from the source image to a standardized image (texture), and then to the target image. Information flow is introduced by X2Face to map from the output given driving information to the learned embedded face. Although these techniques are target-independent, they are still unable of producing photo-realistic images of faces. Artifacts such as mismatched facial details and blurriness frequently show up in the altered outputs. And due to the introduction of the warping process, it produces blurry results when dealing with extreme poses and expressions. X2Face is one of the earliest learning-based techniques for animating human heads that doesn't require any prior facial information. In some instances, their warping technique results in strange head deformations, which results in poor photorealism.

Monkey-Net (Siarohin, Lathuiliere, et al., 2019) predicts numerous pairs of unsupervised key points to estimate optical flow for animating. It also learns a collection of unsupervised keypoints to produce animations. A more recent deep learning framework called Monkey Net suggests using keypoint detection to infer motion from driving video. Then, the output is produced using the appearance that was derived from the reference image and motion information. However, Monkey-Net struggles to model how objects near the keypoints alter their look, which results in poor generation quality when the scale of object change is quite high.

In the case of significant object pose changes, this results in poor generation quality. It is the first depth model for unknown object image animation. It creates a dense heatmap from sparse keypoints by extracting target keypoints from the image using a self-supervised keypoint detector. The input frame is then created using the motion heatmap and appearance data that was taken from the input image. However, when the scale of object change is quite large, Monkey-Net struggles to represent the appearance modification of objects near the keypoints, which results in poor generation quality.

First Order Motion Model (FOMM) (Siarohin, Lathuilière, et al., 2019) enhances single image animation's outcomes noticeably. The object in the first frame of the driving video must be in the same pose as the one in the source image since FOMM employs relative keypoint locations to protect the identity of the source, which prevents it from animating the object using any annotations or prior knowledge about it. To solve this problem, they suggest modeling complex motions using a set of self-learned keypoints and local affine transformations. Because the warping-based head posture of generated samples is not always guaranteed to follow the driver, which leads to poor generation quality occurs when there are major changes in object pose.

(Zakharov et al., 2019) proposed a few-shot talking heads employs both strategies by first utilizing adaptive normalizations and afterwards adjusting the generator network in the few-shot setting. In most cases, the fine-tuned generator gives a considerably better fit of the training sequence and reduces the identity gap. Nevertheless, these techniques lack in terms of quality and identity preservation when compared to 3D-based subject-specific models. These methods use a meta-learning phase to train a model that is subject-agnostic on a huge corpus of data to close the performance gap. An optional subject-specific fine-tuning phase is then carried out to increase realism and identify the source. The primary drawbacks of these methods are the lack of landmark adaption and the representation of mimics (in particular, the current set of landmarks do not in any way represent the gaze). Using a landmark from someone else causes a noticeable personality mismatch.

Neural head reenactment with latent pose descriptors (Burkov et al., 2020) They propose a head reenactment system driven by latent pose descriptors, unlike other systems that use landmark images or keypoints, and along with the RGB image, it can forecast foreground segmentation. The first addition is the segmentation prediction capability. Second, rather than using keypoints,

the algorithm learns to do reenactment using latent pose vectors. Pose descriptors can be helpful for third-party tasks like emotion recognition because they are person-independent. Pose-identity disentanglement happens automatically, without special losses, because of the constraint on the network's capacity. Their approach struggles to capture some subtle mimics, especially those involving gaze direction, and the segmentation masks level of monitoring is the lowest.

(Hong et al., 2022) developed a self-supervised method for learning pixel-wise depth maps utilizing geometric warping and photometric consistency. The talking-head video creation can gain a number of important advantages from the dense 3D geometry or pixel-level depth. First, the capacity of the model to preserve a realistic 3D face structure is a crucial component for producing high-fidelity face videos since the video captures the moving heads in a realistic 3D physical world, which greatly facilitates an accurate recovery of 3D face structures. Second, particularly in noisy background information, the dense geometry can further support the model in robustly differentiating the noisy background information for generation. It is difficult to statistically evaluate the depth estimation since they lack any ground-truth depths for the facial images.

Siarohin *et al*. proposed motion representations for articulated animation (MRAA) (Siarohin et al., 2021), which improves the shortcomings of FOMM (Siarohin, Lathuilière, et al., 2019) and achieves state-of-the-art performance of unsupervised methods. MRAA employs PCA-based motion estimation, which is more effective at depicting articulated motions (such as those of the human body). To eliminate the negative effects of camera motion, it additionally includes background motion estimation. These unsupervised techniques are capable of performing motion transfer on subjective object. By combining a keypoint and the associated affine matrix into a single heat map estimation, the method suggested in MRAA also enhances the motion learning process. While MRAA struggles to depict relatively inflexible motions (*e.g*., human face) and fails to consider cooperative part motions.

The purpose of this review is to identify major studies related to neural talking heads, their problems, the gap to be filled, and the method used in the system. However, the majority of current methods are target-specific and unable to capture images of previously unseen people. Additionally, they have faults, including blurriness and mismatched facial elements. this study attempts to address the problem that most prior works faced commonly and proposes a

disentangled representation based on one-shot realistic neural talking head synthesis, with a focus on disentangling identity from pose and expression in order to solve the identity mismatch gap, temporal consistency and improve the photorealism of the synthesized faces.

2.8. Summary of Related Work

The list of related works that are illustrated in section 2.7 is summarized in the table below.

Table 2. 1: Summary of related works

Authors	Title	Method	Dataset used	Evaluation metrics	Gap
(Zakharov et al., 2019)	Few-Shot Adversarial Learning of Realistic Neural Talking Head Models	Direct synthesis and supervised	VoxCeleb1 and 2	Qualitative comparison with prior work. Quantitative analysis - SSIM, FID, CSIM and user study.	Landmarks would also carry some identity-related information, resulting in apparent identity gaps in the synthesized face.
(Burkov et al., 2020)	Neural head reenactment with latent pose descriptors	Direct synthesis and supervised	VoxCeleb2	Identity error I_T and Pose reconstruction error P_T.	Some subtle mimics, particularly gaze direction, are difficult for their method to capture.
(Wiles et al., 2018)	X2Face: A network for controlling face generation using images, audio, and pose codes	Warping-based and Autoencoder	VoxCeleb1 LRW dataset	Qualitative and Quantitative comparison with prior work. MAE, L1 loss and comparison with ground truth.	It yields blurry results when dealing with extreme pose and expression. Pose and expression may not be accurately translated since no consideration is given to facial landmark information.

(Siarohin, Lathuiliere, et al., 2019)	Animating Arbitrary Objects via Deep Motion Transfer	Warping-based and unsupervised	UvA-Nemo Tai-Chi BAIR robot pushing dataset	Quantitative and Qualitative analysis. L1 loss, Average keypoint distance (AKD, Missing keypoint rate (MKR), Average euclidean distance (AED) and Frechet inception distance (FID).	Struggles to represent the appearance modification of objects near the keypoints, which results in poor generation quality.
(Siarohin et al., 2019)	First Order Motion Model (FOMM)	Warping-based and motion field estimation	VoxCeleb1 UvA-Nemo Tai-Chi-HD BAIR robot pushing dataset	Quantitative and Qualitative analysis. L1 loss, Average keypoint distance (AKD), Missing keypoint rate (MKR) and Average euclidean distance (AED). Comparison to ground truth (user study).	Relative motion transfer could result in noticeable errors when the driving face and the source face differ in terms of head pose or expressions.
(Hong et al., 2022)	DaGAN++: Depth-Aware Generative Adversarial Network for Talking Head Video Generation	3D facial geometry and self-supervised	VoxCeleb1 VoxCeleb2 HDTF dataset	Structural similarity (SSIM), Peak signal-to-noise ratio (PSNR), L1, Average Keypoint Distance (AKD), and Average Euclidean Distance (AED)	The quality of the depth map can only be evaluated qualitatively through visualization because ground truth depth data is not available.
(Siarohin et al., 2021)	Motion representations for articulated animation	PCA-based motion estimation and unsupervised	VoxCeleb1 Tai-Chi-HD MGif TED-talks	L1 loss, Average keypoint distance (AKD), Missing keypoint rate (MKR) and Average euclidean distance (AED).	Fails to consider cooperative part motions because they use regions for motion estimation.

CHAPTER THREE

3. RESEARCH METHODOLOGY

3.1. Chapter Overview

This section elaborates on the methodologies and approaches utilized for data collection, analysis, and the tools employed in successfully conducting the research. In order to attain the objectives of this study, the following methods and techniques will be applied: dataset selection, data collection procedures, augmentation methods, preprocessing techniques, software and hardware tools, baseline research, and the evaluation metrics utilized.

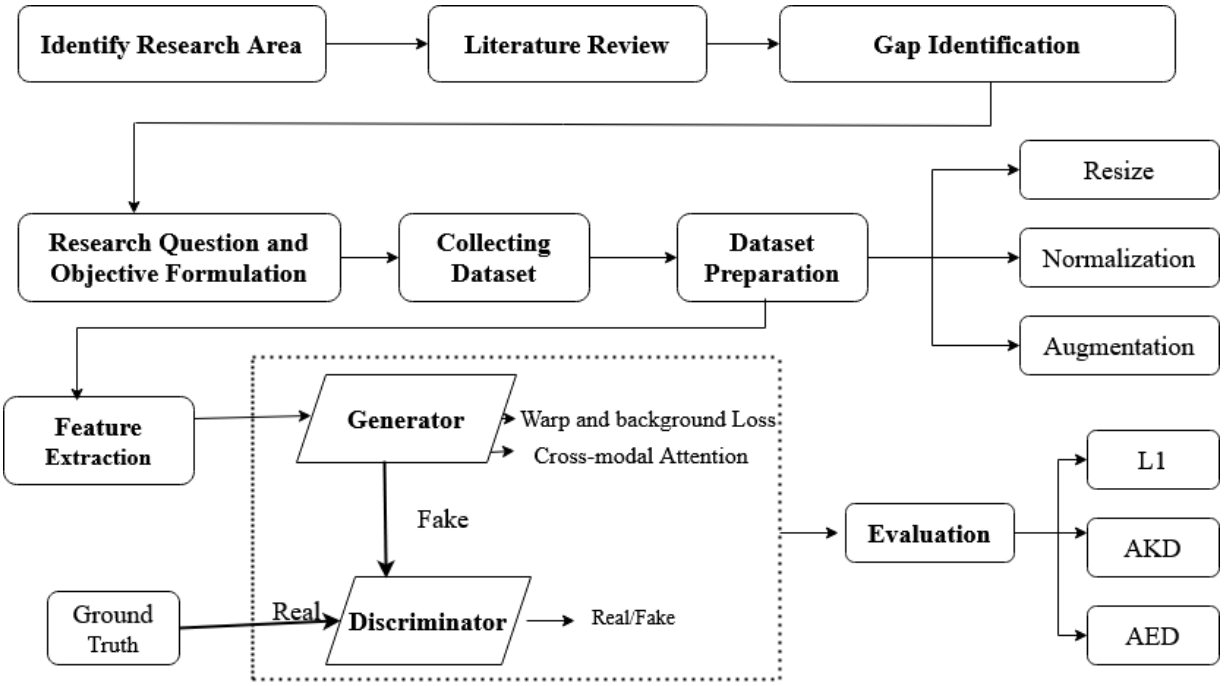


Figure 3. 1: Procedures of the overall research process workflow

3.2. Dataset Collection

The proposed method will be examined and conducted using large scale datasets for both training and testing that are the best appropriate for the issue with photorealistic neural talking head models. For quantitative and qualitative analysis, VoxCeleb1 dataset is used that contains talking head video (Nagraniy et al., 2017) (256p videos at 1 fps).

3.2.1. VoxCeleb1 Dataset

This dataset includes over 100,000 utterances from 1,251 famous people that were taken from YouTube videos. With 55% of the speakers being men, the dataset is gender-balanced. The speakers varied widely in terms of their ages, professions, accents, and races. The nationality and gender of each speaker (obtained from Wikipedia) is also provided (Nagraniy et al., 2017). 12331 training videos and 444 test videos are obtained overall.

Figure 3. 2: VoxCeleb1 dataset samples

3.3. Pre-processing Techniques

Before starting training any deep learning model, the collected raw data are converted to the format that fit the model. Several strategies were employed to preprocess data to build a better dataset.

3.3.1. Resize Image

We take a first frame bounding box from the video for pre-processing. Up until it departs from its starting point too far, we follow this face. Next, we crop the video frames with the smallest crop that still includes all of the bounding boxes. Up until the end of the series, the process is repeated. Sequences with resolutions less than 256 × 256 are removed, and the remaining videos are scaled to 256 × 256 while maintaining the aspect ratio (Siarohin, Lathuilière, et al., 2019).

3.3.2. Normalization

Normalization is the process of transforming features to a consistent scale, which enhances the stability and performance of the model. It mitigates the effects of changes in lighting and contrast within input images, thereby increasing the network's resilience and accuracy. *VoxCeleb1 dataset link: https://www.robots.ox.ac.uk/~vgg/data/voxceleb/vox1.html*

Normalization adjusts the original range of feature values to the standardized range of [-1,1]. The calculation for image normalization is as follows:

$$x = \frac{y - \mu}{\delta} \qquad \text{eq. (3.1)}$$

Here is an input image with pixel values ranging from 0 to 1, and both the mean (μ) and variance (δ) are set at 0.5.

3.3.3. Data Augmentation

The following image augmentation methods were employed: horizontal flipping, adjustments to saturation, contrast, brightness, and hue. Horizontal flipping involves mirroring the image horizontally, producing a realistic result. Saturation and contrast adjustments are applied by randomly altering the image's saturation and contrast within specified ranges. The saturation factor is selected uniformly from the range [max (0, 1 - saturation), 1 + saturation], while the contrast factor is chosen uniformly from [max (0, 1 - contrast), 1 + contrast]. Brightness adjustments are made using a brightness factor of 0.1. The degree of hue variation is determined by calculating the hue factor, which is uniformly chosen from the range [-hue, hue]. This value should be greater than or equal to 0 and less than or equal to 0.5.

3.4. Development Tools

This section provides an overview of the diverse set of development tools utilized in the entire process of designing and implementing the proposed model. In the context of this study, various categories of development tools will be utilized to facilitate the creation of the thesis work. These tools encompass prototype development tools, platforms, UML modeling tools, and other pertinent resources essential for the research. The subsequent sections will offer concise insights into these development tools.

3.4.1. Design Tools

Design tools serve as means for generating, illustrating, and understanding design concepts. In the proposed system, Edraw Max will be employed for design-related tasks. Edraw Max is a 2D software program specifically tailored for business and technical diagramming. It facilitates the creation of various visual representations such as organizational charts, mind maps, network diagrams, floor plans, workflow diagrams, business charts, and engineering diagrams.

3.4.2. Hardware Tools

The research implementation will involve the utilization of the subsequent hardware tools.

Table 3. 1 Hardware Tools

No.	Tools	Used for
1.	GPU	Increasing the computation and accelerating the training process
2.	RAM	Accelerating the training process cooperatively with GPU
3.	Hard Disk	Storing large datasets

3.4.3. Software Tools

Software tools are a collection of computer programs employed for the creation, administration, debugging, or assistance of other applications and programs.

Table 3. 2 Software Tools

No.	Tools	Specification
1.	Anaconda	It is a free and open-source distribution of Python that is primarily concerned with offering everything needed for applications in machine learning and data science. The distribution includes several packages in addition to the built-in Python interpreter.
2.	Jupyter Notebook	It is an open-source web application that includes coding and real-time visualization. It is a web-based program under Anaconda distribution and it let you code Python. You can also call it a web application under Anaconda.

3.	PyTorch	Based on the Python programming language and the Torch library, PyTorch is an open-source machine learning (ML) framework. With TorchScript, PyTorch offers a simple way to switch between eager mode and graph mode as well as a Python package for high-level capabilities like tensor computation (similar to NumPy) with powerful GPU acceleration.

3.5. Baseline Works

In order to assess the effectiveness of the proposed model, it will be compared to a recent approach that utilizes GAN to create a realistic neural talking head. This method's performance has been evaluated across various benchmark datasets. Consequently, for this research, the MRAA architecture introduced by (Siarohin et al., 2021) has been selected as the baseline. This choice is due to MRAA's capability to automatically infer intensity values and deliver impressive results compared to alternative methods. MRAA employs PCA-based motion estimation, which excels in accurately representing complex motions, such as those exhibited by the human body and faces. Furthermore, it incorporates background motion estimation to mitigate the adverse effects of camera movement. These unsupervised techniques are capable of transferring motion to diverse objects.

The MRAA model learns visual animation in an unsupervised manner, using only RGB videos for training and testing with no prior knowledge. Prior to warping source images to produce the final results, they first predicted dense flow fields from the input images. Without any fine-tuning on pre-trained models, inference just needed one image of a target identity. Existing methods continued to use the concept of employing explicit structure representations despite not requiring priors. By substituting a keypoints predictor with a PCA-based region prediction module, they came up with this concept to model articulated objects. Additional networks use regions as intermediate features and train them end-to-end to predict target flow fields. Although the gathering of ground truth labels is more time-consuming than online prediction of such representations, it nevertheless puts a burden on networks' complexity.

MRAA (Motion Representations for Articulated Animation) contains region predictor, background motion predictor, pixel-wise flow predictor and image generator. The region

predictor returns heatmaps for each component in the source and driving images. Better motion segmentation is made possible by increased convergence, more stable, robust object and motion representations, as well as empirically capturing the geometry of the underlying object part. Additionally, by using a background motion predictor, the model may concentrate just on the foreground object, which strengthens the stability of the recognized points and further enhances convergence. The source image is warped in a feature space to create the target image, and the target image is rendered in two steps: first, a pixel-wise flow generator turns coarse motions into dense optical flow, then the encoded features of the source are warped in accordance with the optical flow, and finally, inpainting the missing regions.

The proposed model has some architectural difference with the baseline work that we employ keypoint detector rather than region predictor, this allows the model to handle large pose gaps between the objects in the source and driving images. We also use dense motion network that is similar to the MRAA pixel-wise flow, but dense motion network estimates optical flow and multi-resolution occlusion masks. A multi-occlusion network to repair the details of the picture from multiple scales, and to obtain more accurate results. A cross-modal attention is employed in the decoder of the image generation to facilitate better preservation of facial structures and generation of expression-related micro facial movements. Finally, it is practical to settle that adding disentangled representation to improve image quality, obtain a better temporal consistency and preserve identity-related information.

The objective of this model is to demonstrate the enhancements made by the proposed model in addressing concerns related to the quality, accuracy, and preservation of identity during face reenactment, thanks to the incorporation of specific constraints. The model exhibited its optimal performance when tested on the VoxCeleb1 dataset. Extensively, existing experimental results demonstrated superior performance of MRAA model over the existing cGAN-based methods, including the state-of-the-art one, on the VoxCeleb1 benchmarked dataset.

3.6. Feature Extraction Network

VGG-19 is a convolutional neural network (CNN) model that underwent training using the extensive ImageNet dataset, which comprises more than 14 million images categorized into 1000 classes. VGG-19 consists of 19 layers, comprising 16 convolutional layers responsible for extracting features from input images and 3 fully connected layers for classification purposes.

Image encoders, like deep neural networks such as CNNs, are widely employed for image analysis. Various CNN models have been introduced since their inception, with VGG-19 being one of them. In our context, we utilize VGG-Face CNN descriptors as a global feature extractor based on the architecture of VGG-Very-Deep-19 CNN. Our primary driving loss is based on the perceptual loss (Johnson et al., 2016), calculated using the pre-trained VGG-19 network. Feature extraction is the process of converting the raw pixel values of an image into a high-level representation that captures the semantic content of the image. This feature extraction process is valuable for a range of tasks, including image recognition, image retrieval, assessing image similarity, and generating images. Additionally, feature extraction can simplify and reduce the complexity of input data, facilitating easier processing and analysis.

3.7. Evaluation Metrics

Evaluation metrics play a crucial role in assessing the effectiveness of a deep learning model. When evaluating the performance of the proposed method, both quantitative and qualitative evaluation metrics are employed. The proposed model utilizes a range of comparison metrics to evaluate the photorealism and preservation of identity in the generated images. These metrics include Average Keypoint distance (AKD), L1, and Average Euclidean distance (AED).

3.7.1. L1

In scenarios involving video reconstruction tasks with access to ground truth videos, we assess the quality by measuring the average L1 distance between the pixel values in the generated video frames and those in the ground truth videos. The L1 error, which represents the disparity between the generated and ground-truth videos, is calculated as the mean absolute difference in pixel values between the reconstructed and ground-truth videos.

$$\mathcal{L}_1 = \frac{1}{N}\sum_{i=1}^{N}\|x_i - y_i\| \qquad \text{eq. 3.2)}$$

3.7.2. Average Keypoint Distance (AKD)

We utilize external keypoint detectors to assess whether the motion in the generated video aligns with the motion in the ground truth video. For each frame in both the ground truth and generated videos, we calculate these keypoints. Using these externally computed keypoints, we

determine the Average Keypoint Distance (AKD), which represents the average distance between the detected keypoints in the ground truth and the generated video frames. AKD is computed by averaging the distances between corresponding keypoints in the ground truth and generated videos.

$$\boldsymbol{AKD} = \frac{1}{N}\sum_{i=1}^{N}\sqrt{(x_i - x_k)^2 + (y_i - y_k)^2} \qquad \text{eq. (3.3)}$$

3.7.3. Average Euclidean distance (AED)

Assesses the degree to which the identity is preserved in the reconstructed video by calculating a feature-based metric. This metric involves computing the Average Euclidean Distance (AED) between a feature representation extracted from the ground truth video and the generated video frames. The choice of feature embedding for this metric is specifically designed to evaluate the preservation of identity in the generated content.

$$\boldsymbol{AED} = \sum_{i=1}^{n}\|X_i - Y_i\|^2 \qquad \text{eq. (3.4)}$$

CHAPTER FOUR

4. PROPOSED ARCHITECTURE

4.1. Chapter Overview

This chapter introduces the Disentangled Representation-Based One-Shot Realistic Neural Talking Head Synthesis architecture, which aims to enhance the quality and consistency of image sequences in the context of neural talking head or image animation. The goal is to generate videos that exhibit improved consistency while preserving the identities of the source images. This chapter is structured into three main sections. The first section provides an overview of the proposed model's architecture. The second section delves into the functions involved in model learning, and the final section explains various optimization loss functions and provides training pseudocode.

4.2. Proposed Model Architecture

The proposed model, known as the "Disentangled Representation Based One-Shot Realistic Neural Talking Head Synthesis," is designed to tackle the task of creating realistic video sequences that maintain consistency while transferring the motion observed in a driving video to a static object depicted in the source image. The architecture of this model is directly inspired by the frameworks described in two prior works: "First Order Motion Model for Image Animation" (Siarohin, Lathuilière, et al., 2019) and "Motion Representations for Articulated Animation" (Siarohin et al., 2021). Fundamentally, the aim of this model is to produce video sequences in which an object from a source image is animated to align with the motion observed in a driving video.

The proposed model consists of two primary components: motion estimation and image generation. Within the motion estimation part, there are two sub-components: coarse motion estimation and dense motion prediction. To improve the performance of the model, several modifications have been made. Firstly, we redefine the way we represent motion by using keypoints instead of the previously employed regions (Siarohin et al., 2021). This change leads to better convergence, increased stability, and more robust representations of objects and motion. It also captures the structure of the parts that make up the object, which helps to separate the object from its motion better. To explicitly account for background or camera motion

between training frames, we predict the parameters of a global, affine transformation that explains non-object-related movements. This adjustment allows the model to focus exclusively on the foreground object, enhancing the stability of identified keypoints and overall convergence. Secondly, we predict occlusion masks for each layer of warped feature maps. This strategy ensures that feature maps have varying focuses, enabling more efficient feature fusion. Lastly, we introduce auxiliary loss functions to clarify the responsibilities of each module within the model. This encourages the network to generate high-quality images by promoting a clearer division of labor among its components.

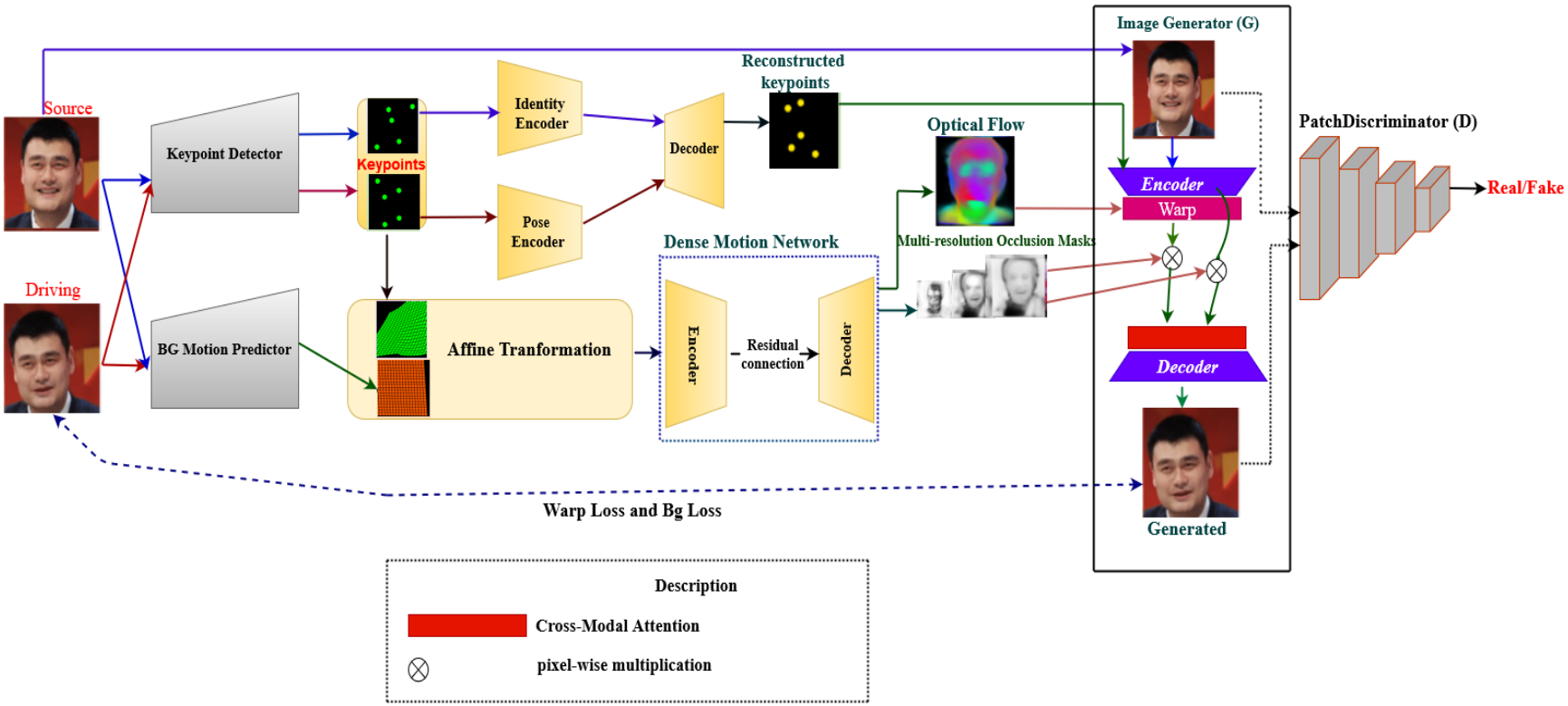


Figure 4. 1: Overall of the Proposed Solution Architecture

4.2.1. Keypoint Detector

The source S and drive D are sent to the keypoint detector, which predicts K × N pairs of keypoints to produce K transformations. With the aid of keypoints that were unsupervisedly learned, we are able to approximate both transformations from sets of sparse trajectories. An encoder-decoder network independently predicts the positions of the keypoints in D and S. For the keypoint detector, we employ the architecture of ResNet18 (He et al., 2016) and adjust the number of neurons in the fully connected layer to K×N×2 for the keypoint detector. The purpose of keypoint detector to predict the coordinates of keypoints for a given number of affine transformations on the object in the source and driving image. The keypoints are then used to estimate an affine transformation motion field that warps the feature maps of the source image to the feature domain of the driving image. This allows the model to handle large pose gaps between the objects in the source and driving images and produce more realistic animations (Zhao & Zhang, 2022). For the keypoint detector, we change the number of neurons in the completely connected layer to K×N×2.

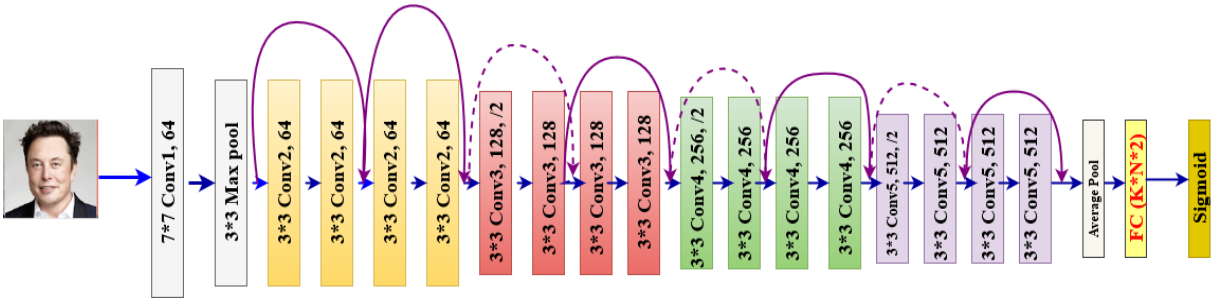


Figure 4. 2: Keypoint Detector Architecture

4.2.2. Background Motion Predictor

The majority of the image is taken up by the background. As a result, even slight background motion between frames, such as that caused by camera movement, degrades the quality of the animation. FOMM (Siarohin, Lathuilière, et al., 2019) does not address background motion independently and represent it using keypoints. The BG motion predictor uses the concatenation of S and D to estimate the affine background transformation's parameters. While the main goal of background modeling is to increase the network's capacity to handle objects more effectively. Background motion estimation is a module that aims to expand the image quality by eliminating

the negative effects of camera motion on the image animation. It uses a ResNet-18 model (Zhao & Zhang, 2022) to predict the affine transformation that represents the background motion from the source image to the driving image and modify the number of neurons in the fully connected layer to 6 neurons. The affine transformation is then used to warp the feature maps of the driving image to the feature domain of the source image. This allows the model to align the background regions of the source and driving images and produce more consistent animations. Background motion estimation can also reduce the noise and blur in the image sequences caused by camera movement.

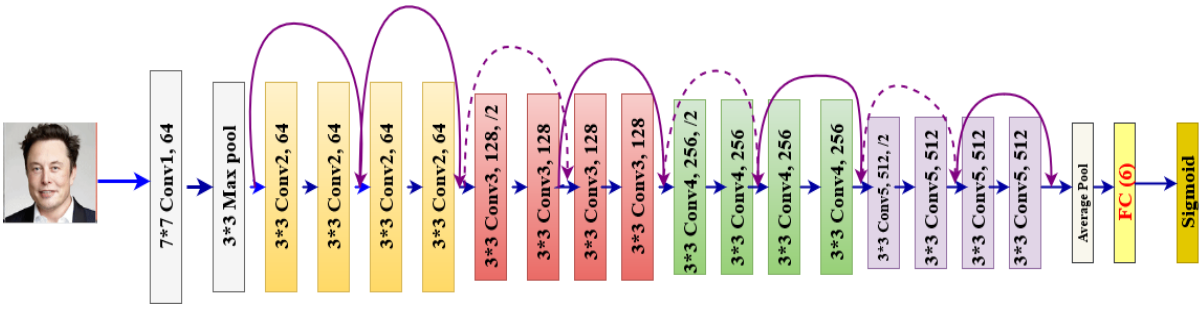


Figure 4. 3: Background Motion Predictor Architecture

4.2.3. Dense Motion Network

In this section, we go into detail about how optical flow estimation and multi-resolution occlusion mask prediction are done. An hourglass network underlies the module. K transformations are received from the keypoint detector, and one affine transformation is received from the background predictor. These K+1 transformations will be combined to estimate the optical flow. The multi-resolution occlusion masks are simultaneously predicted by different layers of the decoder portion to show missing areas of the warped feature maps. First, a motion representation that warps the source image's feature maps to the driving image's feature domain is used to estimate an optical flow and second, an occlusion mask is predicted to show the missing areas of the distorted feature maps, which are subsequently inpainted in the network. In (Siarohin, Lathuilière, et al., 2019) and (Siarohin et al., 2021), for the warped feature maps to fill in the missing regions, a dense motion network predicts a single occlusion mask. However, numerous studies have demonstrated that when the scale increases, the feature map's focus shifts. Detailed textures are more important to high-scale feature maps than abstract patterns are too low-scale feature maps. A single occlusion mask can learn during training how to trade off

the focus of different scale feature maps if it is used to mask out feature maps of various scales. In light of this, we forecast occlusion masks with various resolutions for each layer of the feature map. By adding an additional convolution layer at each decoder layer, the dense motion network will be able to forecast the multi-resolution occlusion masks in addition to estimating the optical flow. The image generation or inpainting network receives both the predicted optical flow and the multi-resolution occlusion masks.

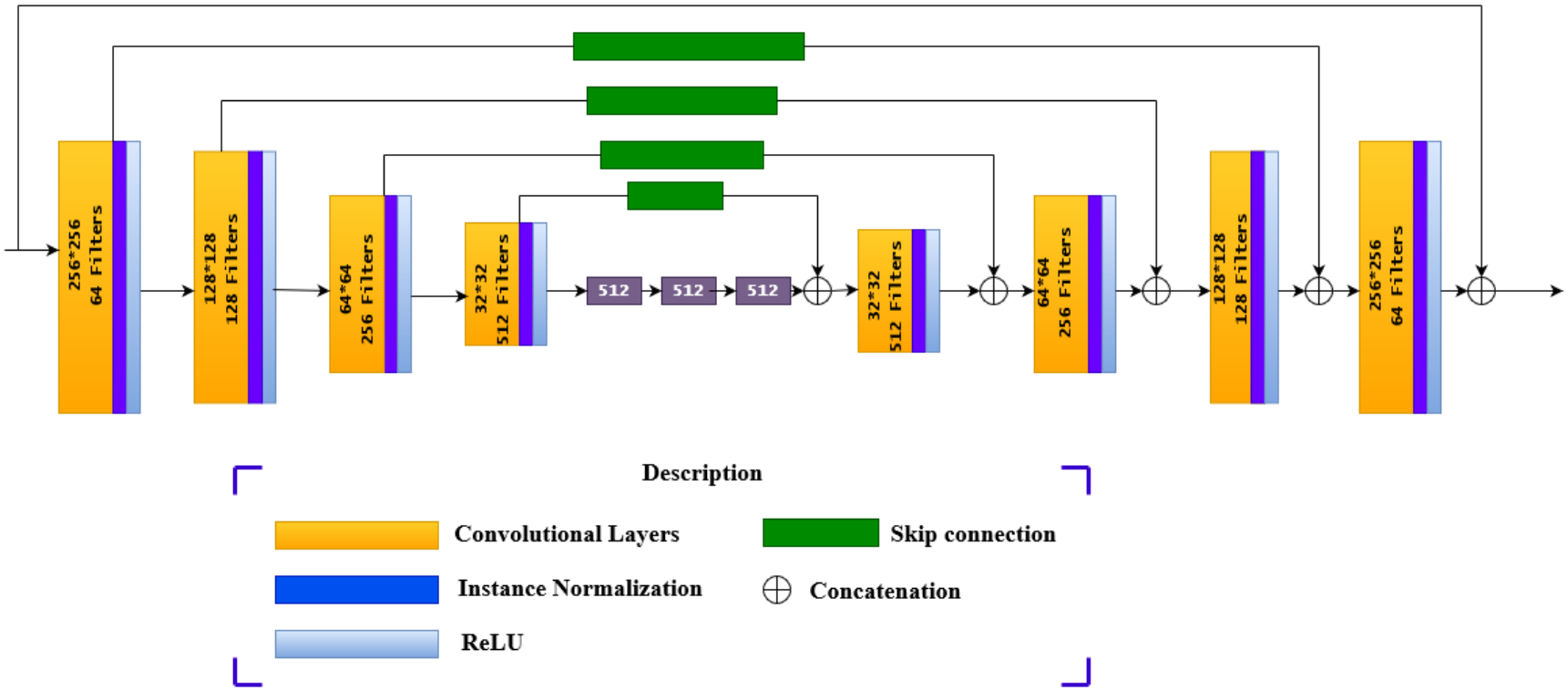


Figure 4. 4: Dense Motion Network Architecture

The implementation defines four blocks that are used in the hourglass architecture:

- ResBlock2d: A residual block that preserves spatial resolution.
- UpBlock2d: An upsampling block used in the decoder.
- DownBlock2d: A downsampling block used in the encoder.
- SameBlock2d: A simple block that preserves spatial resolution.

Downsampling blocks consist of several components:

- Convolutional layers
- Instance normalization
- Rectified linear unit (ReLU) and
- Average Pool

Upsampling blocks comprise elements such as:

- Convolution layers
- Instance normalization and
- Rectified linear unit (ReLU) activation function

4.2.4. Image Generation

Additionally, it is an hourglass network. Using the estimated optical flow, it warps the feature maps of the source image and inpaint in the missing areas of the feature maps for each scale. The final layer outputs the generated image. We use multiple scale features to create high-quality images during the image generating process. S is fed into the encoder, and the feature maps of each layer are warped using optical flow. The warped feature map is then concatenated to the decoder component through a skip connection and masked using the occlusion mask of the appropriate resolution. After passing through two residual blocks, the feature map is then upsampled. At the last layer, the generated image is output. However in contrast to FOMM (Siarohin, Lathuilière, et al., 2019), but similar to Monkey-Net (Siarohin, Lathuiliere, et al., 2019), here deformable skip connections (Siarohin et al., n.d.) are used between the encoder and decoder.

Therefore, to summarize the components within each block:

In other words, downsampling blocks consist of the following elements:

- Convolutional layers
- Instance normalization
- Rectified linear unit (ReLU) and
- Average Pool

Residual blocks comprise the following components:

- Convolution layers
- Instance normalization and
- ReLU activation function.

Upsampling blocks consist of the following elements:

- Convolution layers
- Instance normalization and
- ReLU activation function.

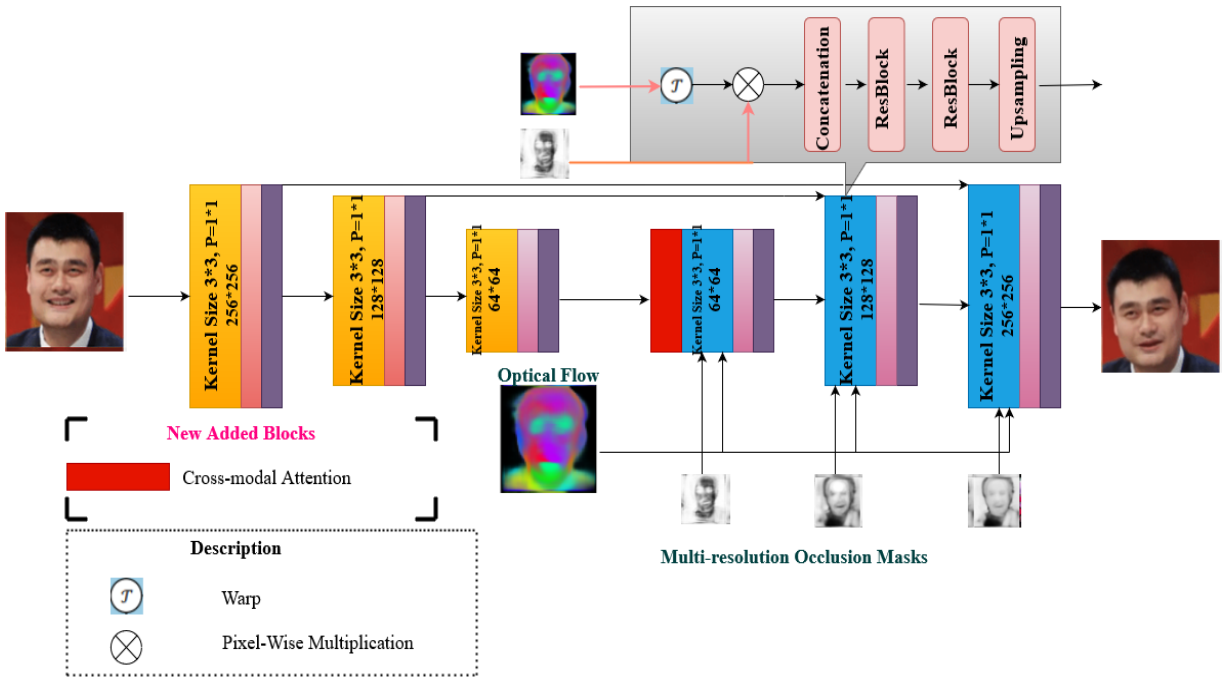


Figure 4. 5: Generator Architecture

4.2.5. Cross-Modal Attention Mechanism

Cross-modal attention is a technique that aims to learn the correspondence and similarity between different modalities or domains of visual data, such as images, text, videos, and 3D shapes. Cross-modal attention can help to fuse and align the features from different sources and selectively attend to the most relevant regions or words for a given task. Cross-modal attention has been applied to various computer vision tasks, such as image-text matching, talking face synthesis, facial expression transfer, lip reading, multi-label image classification, image retrieval, action recognition, and 3D vision. It also used in generative adversarial networks (GANs) can help to generate realistic and diverse data from one modality conditioned on another modality, such as text-to-image synthesis, image-to-image translation, and image inpainting. Cross-modal attention in GANs can exploit the inter-modality and intra-modality correlations and similarities by using attention mechanisms in both the generator and the discriminator networks. Cross-modal attention in GANs can improve the quality and diversity of the generated data and the discriminability and robustness of the learned representations.

The cross-modal attention mechanism, which enhances expression-related micro-movements while reducing background noise motion, intends to accelerate learning of the facial motion field. The estimated motion field, the keypoint of the image, the N encoded feature maps $\{\mathbf{F}_e^i\}_{i=1}^N$, which are extracted by a CNN encoder with the source image $\mathbf{I}_S$ as input, and the output enhanced feature maps are used to create the final output image during the geometry enhanced multi-layer generation process. We also learn cross-modal attention using the keypoint map and the warped source feature map $\mathbf{F}_w^i$ which is the output of the motion flow $\boldsymbol{A}_{S\leftarrow D}$ by warping of the source feature map $\mathbf{F}_e^i$ to encourage the model to keep facial structure details in each layer. With geometry information incorporated into the generating process, we can provide output with high-quality structure preservation.

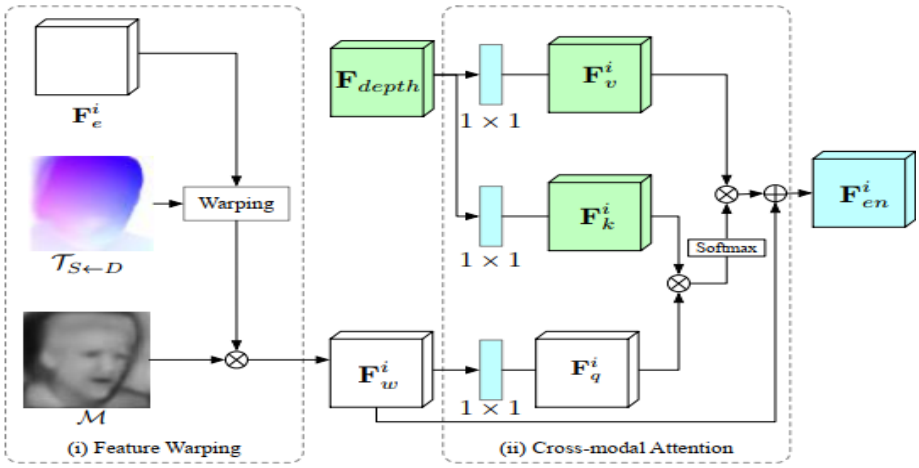


Figure 4. 6: Cross-modal attention GAN
(Source: adapted from ((Hong et al., 2022)))

As seen in Figure 4.5, a linear projection is performed on both $\mathbf{F}_{keypoint}$ and the warped source-image feature $\mathbf{F}_w^i$, resulting in $\mathbf{F}_q^i$, $\mathbf{F}_k^i$, and $\mathbf{F}_v^i$ using three distinct 1×1 convolutional layers with kernels $\mathbf{W}_q^i$, $\mathbf{W}_k^i$, and $\mathbf{W}_v^i$, respectively. The cross-attention mechanism's query, key, and value are represented by these maps, $\mathbf{F}_q^i$, $\mathbf{F}_k^i$, and $\mathbf{F}_v^i$ respectively. To create the query $\mathbf{F}_q^i$, which has the same shape as the output, we use the appearance feature map $\mathbf{F}_w^i$. As a result, using cross-attention from the geometry features, geometry-related information can be accessed. By applying an addition operation, the geometry-related data is also added to the appearance feature map through the residual connection. Thus, dense guidance for face generation can be provided by the keypoint features. In details, the geometry-enhanced features $\mathbf{F}_{en}^i$ for generation at the $\boldsymbol{i}^{th}$ layer are obtained as follows:

$$\mathbf{F}_{en}^i = \text{Softmax}\,(\mathbf{F}_q^i \otimes (\mathbf{F}_k^i)^T) \otimes \mathbf{F}_v^i + \mathbf{F}_w^i, \qquad \text{eq. (4.1)}$$

where Softmax (·) is a softmax operation, and ⊗ indicates a matrix multiplication operation. Cross-modal attention can improve the performance and robustness of these tasks by exploiting the complementary information and reducing the noise across modalities.

The proposed architecture's generator is made up of encoder that warps features of the source image and decoder that output feature maps for generating the final output image. When the encoder produces ghosted faces during the synthesis process, the decoder can detect such anomalies and force that generator to produce satisfied faces via skip connection. In a way, the decoder provides an attention mechanism for the earlier ones to talking head effectively. Hence, the cross-attention layer was used in the generator architecture of the proposed work before the upsampling block to focus on these areas of the face image relevant to talking head through the decoder as shown in figure 4.5.

A better-generated image can be produced by using the cross-modal attention map to refine the distorted feature and create a refined feature. To provide finer-grained features of facial structure and movement with more accuracy, we learn cross-modal attention maps using keypoint maps to constrain the motion field. We present a cross-modal (i.e., keypoint and image) attention method, allowing improved preservation of face structures and creation of expression-related micro facial movements, in order to effectively use the obtained facial keypoint maps for improving generation quality. The cross-modal attention layer is incorporated into the initial decoder layer of the image generator, where it takes in the feature information derived from the final convolutional layer.

4.2.6. Talking Head Synthesis via Disentanglement

Standard and relative methods for image animation both have drawbacks. While relative animation can be used with a smaller set of inputs, such as the requirement that objects be in the same pose in both the source **S** and initial driving **D1** frames, the standard method simply transfers object shape or appearance from the driving frame into the produced videos. In order to overcome this, we develop shape or appearance and pose or expression encoders, which use a separate trained network to predict the motion that should be applied to **S**. As in MRAA, we train an identity and a pose encoder (Siarohin et al., 2021). The pose encoder learns the pose of the keypoints of **Dt**, whereas the shape encoder learns the shape of the keypoints of **S**. The keypoints are then rebuilt by a decoder while keeping **S** identity and **Dt** pose. Encoder and decoder implementation uses fully connected layers. In order to train the networks, we use the keypoints of two frames from a video, with the keypoints of one frame randomly modified to simulate the pose of a different identity. L1 loss is applied to motivate networks to reconstruct

the keypoints prior to deformation. We feed the keypoints of **S** and **Dt** through the shape and pose encoders to obtain the reconstructed keypoints for image animation.

Fully connected layers are used to implement the encoders and the decoder, which are trained independently of other blocks using an L1 reconstruction loss on the motion parameters. The object has the same shape as the source and driving frames from earlier training, thus to confirm that shape originates from the right branch, random horizontal and vertical scaling deformations are added to the driving motions during training. The other branch shape must now provide shape information as a result. The pose must still originate from the pose branch, despite the fact that the shape branch has a different pose. Thus, three Linear - BatchNorm1D - ReLU blocks and a linear layer with 64 neurons make up both the shape or identity and pose encoders. The decoder receives the concatenation of the two latent feature maps from these two encoders, which comprises of a linear layer of K×N×2 neurons and three Linear-BatchNorm1D-ReLU blocks with sizes of 1024, 512, and 256. The target keypoints are then generated by a decoder while maintaining the motion from the driving keypoint and the shape or identity from the source keypoint.

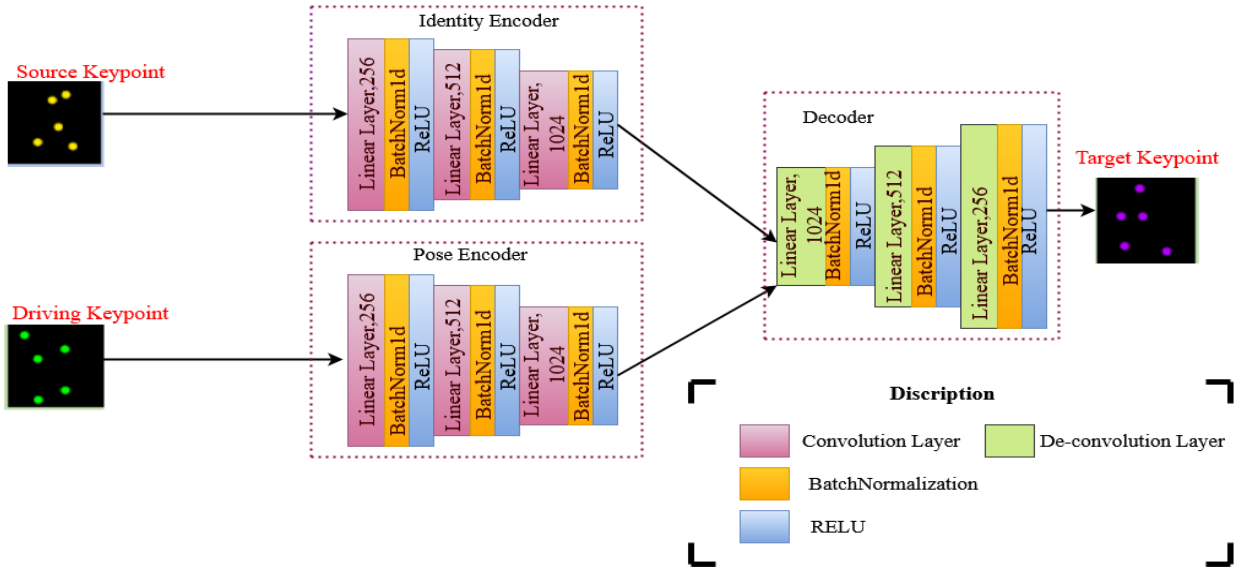


Figure 4. 7: Disentangled Representation Architecture

4.2.7. Discriminator Network

The discriminator network employed here follows the PatchGAN architecture introduced in the (Isola et al., 2017). This approach has demonstrated remarkable success in various image-to-

image translation tasks, including neural talking head synthesis. PatchGAN is a specific type of discriminator used in Generative Adversarial Networks (GANs). Its key characteristic is that it penalizes the structure of local image patches, rather than the entire image. In other words, it assesses whether each small N × N patch within an image is real or fake. The overall output of the discriminator is obtained by applying this patch-wise classification across the entire image and averaging the responses. The PatchGAN discriminator operates under the assumption that pixels separated by more than the diameter of a patch can be considered independent. This modeling approach effectively treats the image as a Markov random field and can be viewed as a form of texture or style loss. PatchGAN is particularly useful in image-to-image translation tasks, such as inpainting, style transfer, and domain adaptation. It often yields sharper and more realistic images compared to a global discriminator that evaluates the entire image as a whole, as demonstrated by (Demir & Unal, 2018).

The discriminator adopted in the proposed architecture contains the following structures:

- Convolution
- Spectral normalization
- LReLU activation function

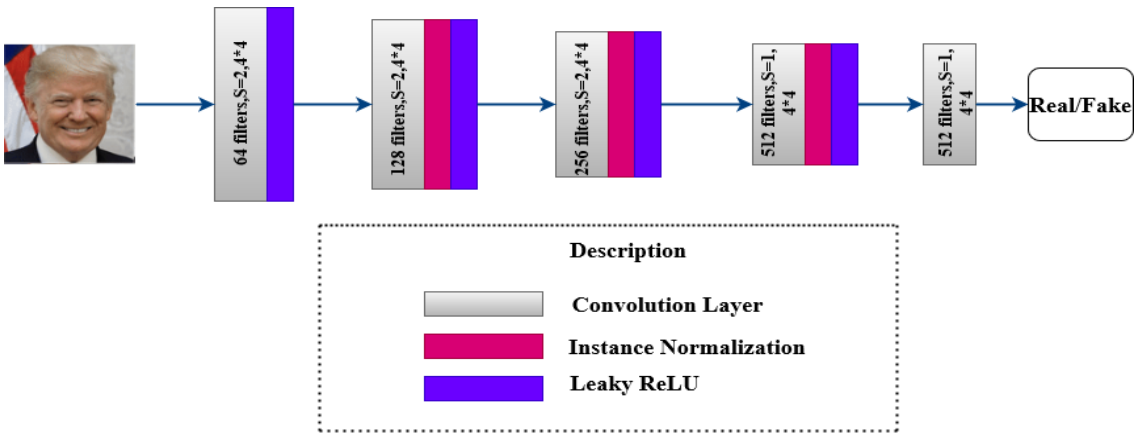


Figure 4. 8: Discriminator Architecture

4.3. Model Learning Functions

Designing an effective learning function is widely recognized as a crucial factor in achieving excellent outcomes. In the context of Generative Adversarial Networks (GANs), the loss

function serves as a critical tool to measure the disparity between the generated image and the genuine image, serving as the optimization objective. When the loss function yields a smaller value, it indicates a smaller gap between the generated and real images, leading to a more impressive reconstruction result. A well-crafted learning function provides precise gradient information for training the network, thereby enhancing the overall performance of image reconstruction.

To achieve the main goal of this study different learning functions like loss function is introduced on the base MRAA (Siarohin et al., 2021) and FOMM (Siarohin, Lathuilière, et al., 2019) for a better quality image animation. Therefore, the proposed model is rooted in the fundamental concept behind it. The overall loss functions used for MRAA (Siarohin et al., 2021) and FOMM (Siarohin, Lathuilière, et al., 2019) comprise two components to meet three requirements of image animation: perceptual loss $\mathcal{L}_{rec}$ and equivariance loss $\mathcal{L}_{eq}$. So, the total loss used in this work is as follows:

$$\mathcal{L} = \mathcal{L}_{rec} + \mathcal{L}_{eq} \qquad \text{eq. (4.2)}$$

4.3.1. Perceptual Loss

We calculate the reconstruction loss between D and the generated image $\widehat{\boldsymbol{D}}$ at multi-resolutions using a pretrained VGG-19 network (Johnson et al., 2016) in the same manner as FOMM and MRAA:

$$\mathcal{L}_{rec} = \sum_{j} \sum_{i} |V_{i(D_j) - V_i}(\widehat{Dj})| \qquad \text{eq. (4.3)}$$

where V_i is the i^{th} layer of the pre-trained VGG-19 network, and j denotes that the image is downsampled j times.

4.3.2. Equivariance Loss

In both FOMM and MRAA, the keypoint detector is additionally constrained by equivariance loss:

$$\mathcal{L}_{eq} = |\boldsymbol{E}_{kp(\mathcal{T}_{ran}(\boldsymbol{S}))} - \mathcal{T}_{ran}\left(\boldsymbol{E}_{kp}(\boldsymbol{S})\right)| \quad \text{eq. (4.4)}$$

where $\mathcal{T}_{ran}$ is the random nonlinear transformation.

4.3.3. Background Loss

To simulate the background motion, we employed an affine transformation, and we added constraints to the BG motion predictor to increase the predictor's accuracy and stability. To produce the affine transformation matrix $\boldsymbol{A}_{bg}$, which represents the background's motion from **S** to **D**, we cascade in the order of **S** and **D** and feed them into the BG motion predictor. After that, we re-cascade them in the reverse direction to get the affine transformation matrix A'_{bg}. The two affine transformation matrices should continue to be consistent, as expected:

$$\begin{bmatrix} A'_{bg} \\ 0\ 0\ 1 \end{bmatrix} = \begin{bmatrix} A_{bg} \\ 0\ 0\ 1 \end{bmatrix}^{-1} \quad \text{eq. (4.5)}$$

Although it is simple to have the BG motion predictor generate a zero matrix that reduces the difference between the two sides of the equation, we are unable to use eq. 4.5 as a loss function. Equation 4.5 is reformulated as follows:

$$\mathcal{L}_{bg} = \left| \begin{bmatrix} \boldsymbol{A}'_{\boldsymbol{bg}} \\ \boldsymbol{0}\ \boldsymbol{0}\ \boldsymbol{1} \end{bmatrix} \begin{bmatrix} \boldsymbol{A}_{\boldsymbol{bg}} \\ \boldsymbol{0}\ \boldsymbol{0}\ \boldsymbol{1} \end{bmatrix} - \boldsymbol{I} \right| \quad \text{eq. (4.6)}$$

where I is 3×3 identity matrix.

4.3.4. warp loss

For image generation, there is still another constrain are designed. We supply **D** to the inpainting network's encoder. With the encoder feature maps of **D** at each layer, the loss is calculated using the warped encoder feature maps of **S**:

$$\mathcal{L}_{warp} = \sum_{i} \left| \widetilde{\mathcal{T}}\left(\boldsymbol{E}_i(\boldsymbol{S})\right) - \boldsymbol{E}_i\,(\boldsymbol{D}) \right| \quad \text{eq. (4.7)}$$

where $\boldsymbol{E_i}$ is the $\boldsymbol{i^{th}}$ layer of the image generator network encoder. The network can estimate the optical flow more accurately with the help of $\boldsymbol{\mathcal{L}_{warp}}$, bringing the warped feature maps closer to the feature domain of D.

4.3.5. Final loss

In order to produce a generate high-quality images or a framework for motion-driven image animation that automatically creates videos by fusing motion patterns taken from a driving video with appearance information from a source image (e.g., encoding the facial expressions of another person), The final loss is the sum of terms is computed as follows:

$$\boldsymbol{\mathcal{L} = \mathcal{L}_{rec} + \mathcal{L}_{eq} + \mathcal{L}_{bg} + \mathcal{L}_{warp}} \qquad \text{eq. (4.8)}$$

4.4. Training Pseudocode

This section outlines the training methods employed in this study. The entire training process in this study is streamlined by following the pseudocode provided in the algorithm below. Consequently, during each iteration, a face image is randomly chosen from the respective dataset. Algorithm 1 was utilized to train the realistic neural talking head using the GAN model in an unsupervised fashion. The entire setup and training procedure are designed to minimize the gap between the authentic ground truth and the generated face image. The bold text signifies the incorporation of additional learning functions.

Algorithm 1: The overall training algorithm of the proposed model.
Input: Source image ($\boldsymbol{S_I}$) and Driving video ($\boldsymbol{D_V}$). **Output:** Animated video of the source image ($\boldsymbol{S_V}$). 1. Preprocessing the dataset, such as cropping, aligning, resizing. 2. Split the driving video into frames. 3. Train discriminator (D): 3.1. Define a loss function, such as perceptual loss, to measure the difference between the model's output and the ground truth image.

3.2. Calculate discriminator perceptual loss from output image with the ground truth image.

3.3. Backpropagate the loss and update the model's parameters with the optimizer.

4. Train generator (G):

4.1. The warped feature map is concatenated to the decoder component through a skip connection after being masked with the occlusion mask of the matching resolution.

4.2. Warp loss is added to the image generator to measures the difference between the encoder feature maps of the driving image and the decoder feature maps of the generated image.

4.3. Background loss is added to measures the consistency between the background parameters predicted from source to driving and from driving to source.

4.4. Add cross-modal attention to the proposed model for the generator decoder.

4.5. Reconstruction loss is calculated from the source image and synthesized ones.

4.6. Update generator parameters to minimize reconstruction loss.

5. Finally, get an identity-preserved source image.

CHAPTER FIVE

5. IMPLEMENTATION OF THE PROPOSED MODEL

5.1. Chapter Overview

This chapter provides a thorough explanation of how the proposed model is put into practice. It covers various aspects such as setting up the environment, preparing the data, and conducting the training process. Additionally, it includes detailed explanations and examples of code snippets used in the implementation.

5.2. Working Environments

This section outlines the setup of the environment and provides detailed information about the hardware specifications utilized during the model's implementation.

- Laptop:
  - Operating System: Windows 10
  - Processor: Intel® Core™ i7-7500U CPU @ 2.70GHz
  - Graphics: Intel ® HD Graphics 620 and AMD Radeon™ R7 M340
  - Installed memory (RAM): 8.00 GB
  - System Type: 64-bit operating system, x64-based processor
- Desktop:
  - Operating System: Ubuntu 20.04.2 LTS
  - Processor: Intel® Core™ i5-4590 CPU @ 3.30GHz x 4
  - Graphics: NVIDIA GeForce RTX 3060 GPU
  - Installed memory (RAM): 12.00 GB
  - System Type: 64-bit

5.3. Environment Setup

In the course of this study, a variety of software tools and Integrated Development Environments (IDEs) were employed.

Anaconda: This is a free and open-source Python distribution primarily designed to cater to the needs of data science and machine learning applications. It is employed for configuring Python

and its assorted modules. Anaconda version 1.9.7 is used for installing essential libraries and modules.

Jupyter Notebook: This is an open-source web application that includes real-time visualization and coding capabilities.

Sublime Text: It is a cross-platform shareware source code editor that has a Python application programming interface (API) and native support for a variety of coding and markup languages.

5.4. Training Procedure

To accomplish the overall objectives of this study, the following processes carried out.

- Dataset collection.
- Data preprocessing.
- Setting up the Graphics Processing Unit (GPU) and preparing the dataset.
- Executing the entire process, starting from training through to testing.

Therefore, to offer a more comprehensive overview of the completed tasks, a thorough explanation of each phase is supplied.

5.5. Proposed Model Implementation

The proposed model "Disentangled Representation Based One-Shot Realistic Neural Talking Head Synthesis" generating videos automatically by fusing motion patterns taken from a driving video with appearances taken from source images. We utilize the ResNet18 (He et al., 2016) architecture for both the keypoint detector and the BG motion predictor. In the keypoint detector, we adjust the number of neurons in the fully connected layer to K×N×2, while for the BG motion predictor, we set it to 6. We employ an hourglass (T. Wang et al., 2019) architecture for the dense motion network and the image generator network, and their encoders have five and four Convolution – InstanceNorm (Ulyanov et al., 2016) - ReLU - AvgPooling blocks, respectively. The decoder of the dense motion network consists of five "Upsampling - Convolution - InstanceNorm - ReLU" blocks and the decoder of the image generator network.

Table 5. 1 Content of the generator architecture

Generator	
Layers in the encoder	Content of each block
1	Convolution – (filters=64, kernel=(3x3), strides=1, padding = 1), Instance normalization, ReLU
2	Convolution – (filters=128, kernel=(3x3), strides=1, padding = 1), Instance normalization, ReLU
3	Convolution – (filters=256, kernel=(3x3), strides=1, padding = 1), Instance normalization, ReLU
4	Convolution – (filters=512, kernel=(3x3), strides=1, padding = 1), Instance normalization, ReLU
Layers in the decoder	
1	Deconvolution – (filters=512, kernel=(3x3), strides=1, padding = 1), Instance normalization, ReLU
2	Deconvolution – (filters=256, kernel=(3x3), strides=1, padding = 1), Instance normalization, ReLU
3	Deconvolution – (filters=128, kernel=(3x3), strides=1, padding = 1), Instance normalization, ReLU
4	Deconvolution – (filters=64, kernel=(3x3), strides=1, padding = 1), Instance normalization, ReLU

Table 5. 2 Content of the discriminator architecture

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

5.5.1. Keypoint Detector Implementation

As discussed in section 4.2.1 a keypoint detector receives source S and driving D to estimate K×N pairings of keypoints and produce K transformations. The amount of keypoints required to show that K×5 pairs of keypoints produce the best motion transfer results. The keypoint detector has a harder time making accurate predictions when more keypoints are predicted. Affine transformations produced by fewer keypoints, on the other hand, are not representing enough to depict motions.

```
class KPDetector(nn.Module):
    def __init__(self, num_tps, **kwargs):
        super(KPDetector, self).__init__()
        self.num_tps = num_tps
        self.fg_encoder = models.resnet18(pretrained=False)
        num_features = self.fg_encoder.fc.in_features
```

```
        self.fg_encoder.fc = nn.Linear(num_features, num_tps*5*2)
    def forward(self, image):
        fg_kp = self.fg_encoder(image)
        bs, _, = fg_kp.shape
        fg_kp = torch.sigmoid(fg_kp)
        fg_kp = fg_kp * 2 - 1
        out = {'fg_kp': fg_kp.view(bs, self.num_tps*5, -1)}
        return out
```

Sample code 5. 1 Keypoint Detector implementations

5.5.2. Background Motion Predictor Implementation

The BG Motion Predictor is responsible for predicting the affine transformation that characterizes the motion of the background as it transitions from the source image to the driving image. However, when there is camera movement in videos, it can cause the anticipated keypoints to appear in the background region, resulting in inaccuracies in motion estimation. To tackle this issue, we also make an extra forecast, akin to MRAA (Siarohin et al., 2021), to model the background's motion. This allows the model to concentrate exclusively on the foreground object, thereby enhancing the stability of identified points and facilitating improved convergence.

```
class BGMotionPredictor(nn.Module):
    def __init__(self):
        super(BGMotionPredictor, self).__init__()
        self.bg_encoder = models.resnet18(pretrained=False)
        self.bg_encoder.conv1 = nn.Conv2d(6, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3,
3), bias=False)
        num_features = self.bg_encoder.fc.in_features
        self.bg_encoder.fc = nn.Linear(num_features, 6)
        self.bg_encoder.fc.weight.data.zero_()
        self.bg_encoder.fc.bias.data.copy_(torch.tensor([1, 0, 0, 0, 1, 0], dtype=torch.float))
    def forward(self, source_image, driving_image):
        bs = source_image.shape[0]
        out = torch.eye(3).unsqueeze(0).repeat(bs, 1, 1).type(source_image.type())
```

```
        prediction = self.bg_encoder(torch.cat([source_image, driving_image], dim=1))
        out[:, :2, :] = prediction.view(bs, 2, 3)
        return out
```

Sample code 5. 2 BG Motion Predictor implementations

5.5.3. Dense Motion Network Predictor Implementation

Module that estimating an optical flow and multi-resolution occlusion masks from an affine transformation.

```
class DenseMotionNetwork(nn.Module):
    def __init__(self, block_expansion, num_blocks, max_features, num_tps, num_channels,
                 scale_factor=0.25, bg = False, multi_mask = True, kp_variance=0.01):
        super(DenseMotionNetwork, self).__init__()
        if scale_factor != 1:
            self.down = AntiAliasInterpolation2d(num_channels, scale_factor)
        self.scale_factor = scale_factor
        self.multi_mask = multi_mask
        self.hourglass = Hourglass(block_expansion=block_expansion,
in_features=(num_channels * (num_tps+1) + num_tps*5+1),
                                   max_features=max_features, num_blocks=num_blocks)
        hourglass_output_size = self.hourglass.out_channels
        self.maps = nn.Conv2d(hourglass_output_size[-1], num_tps + 1, kernel_size=(7, 7),
padding=(3, 3))
        if multi_mask:
            up = []
            self.up_nums = int(math.log(1/scale_factor, 2))
            self.occlusion_num = 4
            channel = [hourglass_output_size[-1]//(2**i) for i in range(self.up_nums)]
            for i in range(self.up_nums):
                up.append(UpBlock2d(channel[i], channel[i]//2, kernel_size=3, padding=1))
            self.up = nn.ModuleList(up)
            channel = [hourglass_output_size[-i-1] for i in range(self.occlusion_num-
self.up_nums)[::-1]]
```

```
        for i in range(self.up_nums):
            channel.append(hourglass_output_size[-1]//(2**(i+1)))
        occlusion = []
        for i in range(self.occlusion_num):
            occlusion.append(nn.Conv2d(channel[i], 1, kernel_size=(7, 7), padding=(3, 3)))
        self.occlusion = nn.ModuleList(occlusion)
    else:
        occlusion = [nn.Conv2d(hourglass_output_size[-1], 1, kernel_size=(7, 7), padding=(3, 3))]
    self.occlusion = nn.ModuleList(occlusion)
```

Sample code 5. 3 Dense Motion Network Predictor implementations

5.5.4. Image Generation Network Implementation

In this study, inpaint or image generation is used to inpaint the lost sections and reconstruct the driving image.

```
class InpaintingNetwork(nn.Module):
    def __init__(self, num_channels, block_expansion, max_features, num_down_blocks, multi_mask = True, **kwargs):
        super(InpaintingNetwork, self).__init__()

        self.num_down_blocks = num_down_blocks
        self.multi_mask = multi_mask
        self.first = SameBlock2d(num_channels, block_expansion, kernel_size=(7, 7), padding=(3, 3))
        down_blocks = []
        up_blocks = []
        resblock = []
        for i in range(num_down_blocks):
            in_features = min(max_features, block_expansion * (2 ** i))
            out_features = min(max_features, block_expansion * (2 ** (i + 1)))
            down_blocks.append(DownBlock2d(in_features, out_features, kernel_size=(3, 3), padding=(1, 1)))
```

```
            decoder_in_feature = out_features * 2
            if i==num_down_blocks-1:
                decoder_in_feature = out_features
            up_blocks.append(UpBlock2d(decoder_in_feature, in_features, kernel_size=(3, 3),
padding=(1, 1)))
            resblock.append(ResBlock2d(decoder_in_feature, kernel_size=(3, 3), padding=(1, 1)))
            resblock.append(ResBlock2d(decoder_in_feature, kernel_size=(3, 3), padding=(1, 1)))
        self.down_blocks = nn.ModuleList(down_blocks)
        self.up_blocks = nn.ModuleList(up_blocks[::-1])
        self.resblock = nn.ModuleList(resblock[::-1])
        self.final = nn.Conv2d(block_expansion, num_channels, kernel_size=(7, 7), padding=(3,
3))
        self.num_channels = num_channels
```

Sample code 5. 4 Dense Motion Network Predictor implementations

5.5.5. Cross-Modal Attention Implementation

As discussed in section 4.2.5 to accurately generate fine-grained details of facial structure and movement, a cross-modal attention layer aids the image generator network in creating images that constrain the motion field. The cross-modal attention layer is incorporated into the initial decoder layer of the image generator, and it takes in the feature extracted from the final convolutional layer.

```
class CrossAttention(nn.Module):
    def __init__(self,in_dim,activation):
        super(DepthAwareAttention,self).__init__()
        self.chanel_in = in_dim
        self.activation = activation
        self.query_conv = nn.Conv2d(in_channels = in_dim , out_channels = in_dim//8 ,
kernel_size= 1)
        self.key_conv = nn.Conv2d(in_channels = in_dim , out_channels = in_dim//8 ,
kernel_size= 1)
        self.value_conv = nn.Conv2d(in_channels = in_dim , out_channels = in_dim ,
kernel_size= 1)
```

```
        self.gamma = nn.Parameter(torch.zeros(1))
        self.softmax  = nn.Softmax(dim=-1) #
    def forward(self,source,feat):
        m_batchsize,C,width ,height = source.size()
        proj_query  = self.activation(self.query_conv(source)).view(m_batchsize,-
1,width*height).permute(0,2,1) # B X CX(N) [bz,32,64,64]
        proj_key =  self.activation(self.key_conv(feat)).view(m_batchsize,-1,width*height)
energy =  torch.bmm(proj_query,proj_key) # transpose check
        attention = self.softmax(energy) # BX (N) X (N)
        proj_value = self.activation(self.value_conv(feat)).view(m_batchsize,-1,width*height)
        out = torch.bmm(proj_value,attention.permute(0,2,1) )
        out = out.view(m_batchsize,C,width,height)
        out = self.gamma*out + feat
        return out,attention
```

Sample code 5. 5 Dense Motion Network Predictor implementations

5.5.6. Background Loss Implementation

In the proposed solution, bg loss is to encourage the network to generate a clear background and avoid artifacts caused by optical flow warping. The bg loss is defined as the L1 distance between the generated image and the driving image in the background region, which is determined by an occlusion mask. The bg loss helps to preserve the background consistency and improve the visual quality of the animation.

```
        if self.bg_predictor and epoch >= self.bg_start and self.loss_weights['bg'] != 0:
            bg_param_reverse = self.bg_predictor(x['driving'], x['source'])
            value = torch.matmul(bg_param, bg_param_reverse)
            eye = torch.eye(3).view(1, 1, 3, 3).type(value.type())
            value = torch.abs(eye - value).mean()
            loss_values['bg'] = self.loss_weights['bg'] * value
        return loss_values, generated
```

Sample code 5. 6 Bg loss implementations

5.5.7. Warp Loss Implementation

In the proposed solution, warp loss is to measure the similarity between the warped feature maps and the driving feature maps**.** The warp loss is defined as the L1 distance between the two feature maps, which encourages the network to learn a more accurate optical flow that can align the source and driving images. The warp loss helps to improve the motion transfer quality and reduce distortion.

```
if self.loss_weights['warp_loss'] != 0:
        occlusion_map = generated['occlusion_map']
        encode_map = self.inpainting_network.get_encode(x['driving'], occlusion_map)
        decode_map = generated['warped_encoder_maps']
        value = 0
        for i in range(len(encode_map)):
            value += torch.abs(encode_map[i]-decode_map[-i-1]).mean()
        loss_values['warp_loss'] = self.loss_weights['warp_loss'] * value
```

Sample code 5. 7 warp loss implementations

5.5.8. Discriminator Network Implementation

In this study, PatchDiscriminator operates on patches of the image and outputs a probability map that indicates how realistic each patch is. The PatchDiscriminator helps to train the generator network to produce more realistic images that match the distribution of the training data.

```
class Discriminator(nn.Module):
    def __init__(self, num_channels=3, block_expansion=64, num_blocks=4,
max_features=512,
                 sn=False, use_kp=False, num_kp=10, kp_variance=0.01, **kwargs):
        super(Discriminator, self).__init__()
        down_blocks = []
        for i in range(num_blocks):
            down_blocks.append(
                DownBlock2d(num_channels + num_kp * use_kp if i == 0 else min(max_features,
block_expansion * (2 ** i)),
```

```
                    min(max_features, block_expansion * (2 ** (i + 1))),
                    norm=(i != 0), kernel_size=4, pool=(i != num_blocks - 1), sn=sn))
        self.down_blocks = nn.ModuleList(down_blocks)
        self.conv = nn.Conv2d(self.down_blocks[-1].conv.out_channels, out_channels=1,
kernel_size=1)
        if sn:
            self.conv = nn.utils.spectral_norm(self.conv)
        self.use_kp = use_kp
        self.kp_variance = kp_variance
    def forward(self, x, kp=None):
        feature_maps = []
        out = x
        if self.use_kp:
            heatmap = kp2gaussian(kp, x.shape[2:], self.kp_variance)
            out = torch.cat([out, heatmap], dim=1)
        for down_block in self.down_blocks:
            feature_maps.append(down_block(out))
            out = feature_maps[-1]
        prediction_map = self.conv(out)


        return feature_maps, prediction_map
```

Sample code 5. 8 Discriminator network implementation

5.6. Hyperparameter Configuration

A hyperparameter is a predefined parameter established prior to the commencement of the learning process. These factors encompass variables such as batch size, the number of training epochs, choice of optimization method, learning rate, and the weighting of the loss function. These parameters are modifiable and have a substantial impact on the efficacy of model training. Optimizers are techniques employed to adjust neural network weights and learning rates in order to minimize losses. In this research, the Adam optimization algorithm was chosen to optimize the model. This algorithm updates network weights iteratively based on training data, serving as a replacement for the traditional stochastic gradient descent (SGD) approach. The selection

of Adam was driven by its simplicity in implementation, minimal memory requirements, and computational efficiency. It proves especially effective when tackling intricate problems involving extensive data or parameters. Another vital hyperparameter is the learning rate (lr), which governs the speed at which the model adapts to the given problem. Tuning this learning rate is essential for achieving better training outcomes, often necessitating an iterative trial-and-error process.

Table 5. 3 Hyperparameter Configuration

No.	**Hyperparameters**	**Values**
1.	Epoch	100
2.	Batch size	4
3.	Learning rate (lr)	2.0e-4
4.	num_repeats	75
5.	Exponential decay rates for the first moment (β_1)	0.5
6.	Exponential decay rates for the second moment (β_2)	0.999
7.	Weight of equivariance loss	10
8.	Weight of warp loss	10
9.	Weight of bg loss	10
10.	Optimizer	Adam

5.7. Experimental Class

To enhance the performance of one-shot realistic neural talking head synthesis, it's crucial to conduct various experiments and assess their outcomes. Four distinct experiments were carried out, as outlined in Table 5.4. The first experiment involved utilizing the baseline approach (MRAA), which employs a pretrained VGG-19 as a feature extractor. The second experiment (MRAA+WL+BGL) marks the initial step of the proposed model, incorporating warp and background loss to bring the warped feature maps closer to the driving feature domain and reduce background noise in motion. In the third experiment (MRAA+WL+BGL+CAM), cross-modal attention was introduced to enhance the generation of expression-related micro facial

movements and better preserve facial structures. Lastly, the fourth experiment (MRAA+WL+BGL+CAM+DR) added disentangled representation to significantly improve source object identity preservation and achieve enhanced temporal continuity in the reconstructed video. To summarize, the table provides an overview of all the conducted experiments.

Table 5. 4 List of experiment class

Notations	**Experiment**	**Dataset used**
MRAA	Baseline work	VoxCeleb1
MRAA+WL +BGL	MRAA+WL with bg loss	VoxCeleb1
MRAA+WL +BGL+CAM	MRAA+WL+BGL with attention mechanism	VoxCeleb1
MRAA+WL +BGL+CAM+ DR	MRAA+WL+BGL+CAM with Disentangled Representation	VoxCeleb1

CHAPTER SIX

6. RESULTS AND DISCUSSION

6.1. Chapter Overview

In this chapter, we will showcase the results of the baseline work and the experiments conducted, providing a thorough examination of the implementation details and the tests conducted. We will present these results using tables, figures, and graphs, and compare them with existing neural talking head architectures. We will also evaluate the performance of our proposed model using the assessment metrics outlined in Chapter 3.

6.2. Result of the Study

6.2.1. Experimental Results

To enhance the quality of the research outcomes, four separate experiments were carried out, each employing identical datasets, hardware, and software configurations to ensure comparability. The results pertaining to the voxceleb1 dataset are presented below.

6.2.2. MRAA

MRAA employs PCA-based motion estimation, which is more effective at depicting articulated motions (e.g., human body). Additionally, they have incorporated background motion estimation to mitigate the adverse impacts of camera motion. These unsupervised methods are capable of performing motion transfer on any object (Zhao & Zhang, 2022). MRAA contains, several components including a region predictor, background motion predictor, pixel-wise flow predictor, and generator. The region predictor produces heatmaps for each element in both the source and driving images. To achieve the transfer of individual regions from the source to the driving frame using a whitened reference frame, they proceed by calculating the principal axes of each heatmap. The pixel-wise flow prediction network merges the transformations applied to both the region and the background. Subsequently, the source image undergoes a feature space warp using pixel-wise flow, resulting in the generation of the target image. Newly introduced regions, as indicated by the confidence map, are then filled in during the inpainting process.

Figure 6. 1: Results of MRAA using the VoxCeleb1 dataset for self-reenactment

Figure 6. 2: Results of MRAA using the VoxCeleb1 dataset for cross-identity

As shown in Figures 6.1 and 6.2, we compare the face synthesis results in Figure 6.1 when the source and driving images are of the same person. We then conduct experiments on the VoxCeleb dataset to take advantage of cross-identity motion transfer in Figure 6.2 when the source and driving images are from different person. It is not very good at maintaining visual details like faces and clothes. Ghosts will show up in the images produced by MRAA for human faces (VoxCeleb). It is noticeable that their technique generates fewer artifacts and captures lower facial components, such as the eye areas.

6.2.3. MRAA+WL +BGL

To solve the above-stated problems the second experiment was carried out. In this experiment, we add warp and background loss. It shows some improvement over the baseline work in terms quality of the image and identity consistency. Also, enhance the stability and accuracy of the predicted parameters as well.

Figure 6. 3: Results of mraa+wl +bgl using the VoxCeleb1 dataset for self-reenactment

Figure 6. 4: Results of mraa+wl +bgl using the VoxCeleb1 dataset for cross-identity

6.2.4. MRAA+WL +BGL+CAM

The introduction of the cross-modal attention module (CAM) significantly enhances the quality of generating micro-movements related to facial expressions in human faces. The model can accurately capture head movement. This result demonstrates that our estimation of facial keypoints may be used to more efficiently construct the motion field between human faces. This shows that the proposed method can capture more subtle facial characteristics, like expression and movements.

Figure 6. 5: Results of mraa+wl +bgl+cam self-reenactment on the VoxCeleb1 dataset

Source Driving MRAA+WL+BGL+CAM

Figure 6. 6: Results of mraa+wl +bgl+cam cross-identity on the VoxCeleb1 dataset

As shown in the figure 6.5 and 6.6 even though there was a slight enhancement according to the evaluation metrics as it is mentioned in section 6.4.1. The CAM can create expressions that are more realistic-looking than the previous implementation (for instance, around the eye areas). And can successfully incorporate the face geometry into the generation process to capture important human facial movement (such as expressions), leading to higher-quality face generation.

6.2.5. MRAA+WL +BGL+CAM+ DR

We humans, are sensitive to temporal discontinuities and subtle artifacts in video. Although any object can be used with the resulting models, the created face sequences are occasionally blurry and lack temporal meaningful. It is challenging to learn talking head only through data-driven methods for a number of reasons, one of which is the coupling of information linked to appearance and motion related information. We combine identity and pose related information by learning disentangled representation in order to solve the aforementioned issues. A talking head sequence will be divided into two complementary representations, one of which will contain identity information and the other will contain pose or expression information.

Finally, the proposed solution of this work MRAA+WL +BGL+CAM+DR constitutes constraints like warp loss, background loss, cross-modal attention mechanism and disentangled representation. We disentangle the shape and pose or expression of objects in order to facilitate talking head generation and avoid leakage of the appearance or shape of the driving object. More specifically, it improved the performance of the previous for preserving identity-related

information, keeping identity-irrelevant information such as background unchanged, and generating high quality images.

Figure 6. 7: Results of mraa+wl +bgl+cam+dr for self-reenactment

Figure 6. 8: Results of mraa+wl +bgl+cam+ dr for cross-identity

As shown in figure 6.7 and 6.8 above, the disentangled representation image animation has good quality for the dataset compared to the previous experiments, and do not leak the identity of the driving video to the source images. The final proposed solution (MRAA+WL +BGL+CAM+

DR) produces enhanced or realistic identity preserving source image and suppresses the ghost artifacts towards a better image quality by using a cross-modal attention mechanism over baseline work, background loss, and warp loss. Another benefit of our model is that the generated video has better temporal continuity.

6.3. Sample Training Log for MRAA+WL +BGL+CAM+ DR

In Figure 6.9, the graph illustrates the generator loss for the proposed model. Initially, during the first epoch, the loss stood at approximately 5.53941. Despite some fluctuations in the loss values, there is a consistent and gradual decline in loss as the training progresses. By the time the model reaches the 100 epochs, the loss has reduced to roughly 1.35032.

In Figure 6.10, the graph depicts the discriminator loss for the proposed model. Initially, during the first epoch, the loss was approximately 129.65790. Although the loss exhibits fluctuations with some ups and downs, it consistently decreases as the training progresses towards the later epochs. By the time the model reaches the 100 epochs, the loss has decreased to about 74.33558.

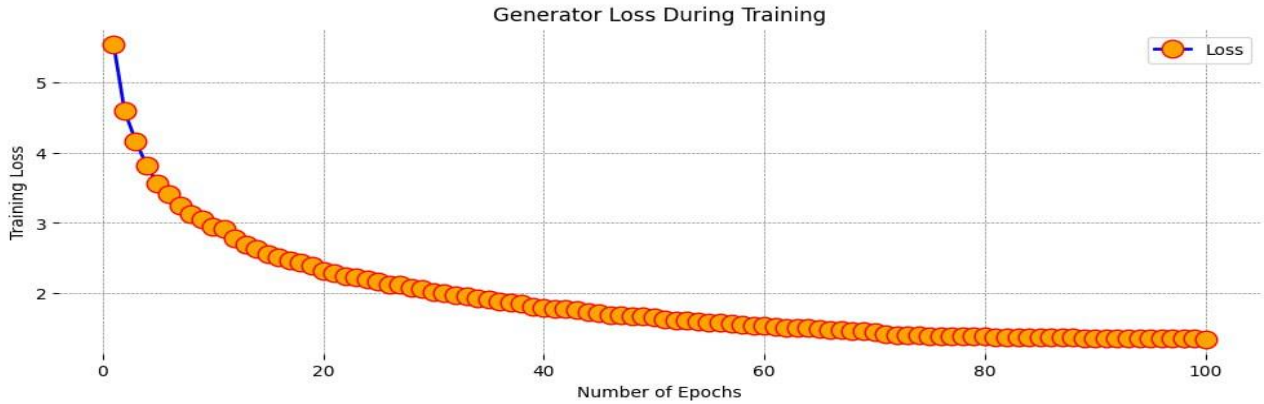


Figure 6. 9: Generator loss graph for MRAA+WL +BGL+CAM+ DR

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

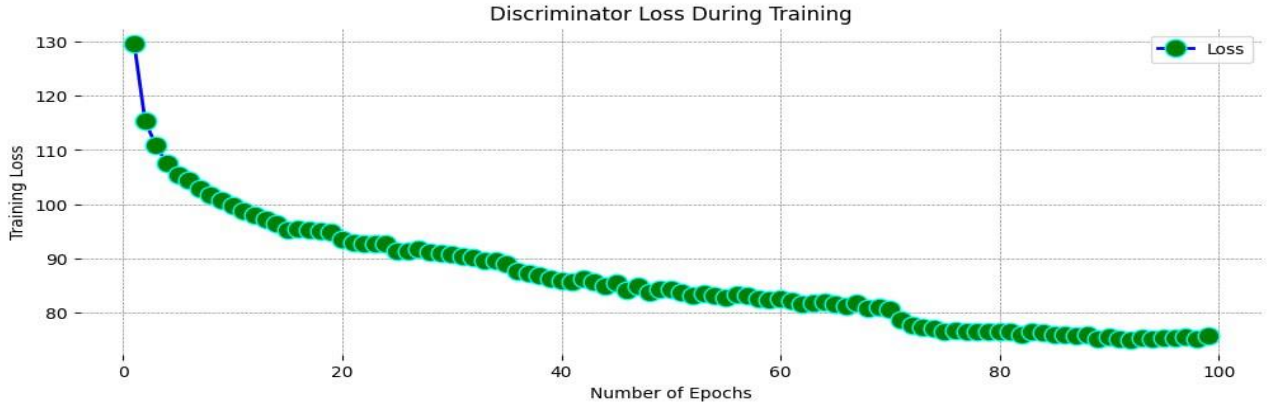


Figure 6. 10: Discriminator loss graph for MRAA+WL +BGL+CAM+ DR

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

6.4. Evaluation Metrics

6.4.1. L1 Error

The various experiments conducted employed L1 metrics to assess the model's performance. L1 was utilized to measure the mean absolute difference between the pixel values of the generated videos and the ground truth. Table 6.1, provided below, showcases the L1 error for all these experiments. A lower score is indicative of better performance, indicating that the GAN is capable of generating images that closely resemble the ground truth. This suggests that the model tends to produce higher-quality facial images. These metrics are calculated by using 444 test videos from the VoxCeleb1 dataset.

Table 6. 1: L1 error for all experiments

No.	Experiments	L1
1	MRAA(Baseline)	0.040
2	MRAA+WL +BGL	0.038
3	MRAA+WL +BGL+CAM	**0.034**
4	MRAA+WL +BGL+CAM+ DR	**0.034**

The bold value indicates the best result in the experiments.

6.4.2. Average Keypoint Distance

The Average Keypoint Distance (AKD) measures the dissimilarity between the poses in the reconstructed video and those in the ground truth video. Landmarks are identified in both videos using face detectors (Bulat & Tzimiropoulos, 2017). AKD is subsequently computed as the mean distance between matching landmarks or keypoints. Table 6.2 displays the AKD values for the conducted experiments, with lower values indicating better results.

Table 6. 2: Average keypoint distance (AKD) for all experiments

No.	Experiments	Average keypoint distance (AKD)
1	MRAA(Baseline)	1.28
2	MRAA+WL +BGL	1.21
3	MRAA+WL +BGL+CAM	1.17
4	MRAA+WL +BGL+CAM+ DR	**1.13**

The bold value indicates the best result in the experiments.

6.4.3. Average Euclidean Distance

Table 6.3 presents a summary of the trained model's ability to recreate images, which is assessed using AED metrics to measure how closely the generated image look like the original one. The Average Euclidean distance (AED) is used to measure the preservation of identity in the reconstructed video. In this process, identity is extracted from both the reconstructed and original frame pairs using publicly available reidentification networks for faces (Amos et al., 2016). Subsequently, we calculate the mean L2 norm of the differences between these identity values across all pairs. Smaller AED values indicate a higher degree of similarity between the generated image and the ground truth image.

Table 6. 3: Average Euclidean Distance (AED) for all experiments

No.	Experiments	Average Euclidean Distance (AED)
1	MRAA(Baseline)	0.133
2	MRAA+WL +BGL	0.125
3	MRAA+WL +BGL+CAM	0.119
4	MRAA+WL +BGL+CAM+ DR	**0.115**

The bold value indicates the best result in the experiments.

6.5. Results Discussion

6.5.1. Discussion on L1 Error Results

To assess the performance of all conducted experiments, L1 metrics were computed and are presented in Table 6.1. Among these experiments, the initial one, considered as the baseline, achieved an L1 score of 0.040. In the second experiment, MRAA employs background and warp loss to train the generator and the discriminators, it shows some improvement over the baseline work in terms of image quality and identity consistency as mentioned in section 6.2.3. This model MRAA+WL+BGL lowers the L1 from 0.040 to 0.038, indicating that the identity quality of the produced image sequences is higher than the prior experiment. In the third experiment, MRAA+WL+BGL+CAM introduced the cross-modal attention layer into the generator network to generate high-quality images. This model scores L1 of 0.034. The last experiment MRAA+WL+BGL+CAM+ DR had the L1 score value of 0.034 which is almost equal to the

prior experiment. It improved the performance of the previous for preserving identity-related information, and generating high quality images.

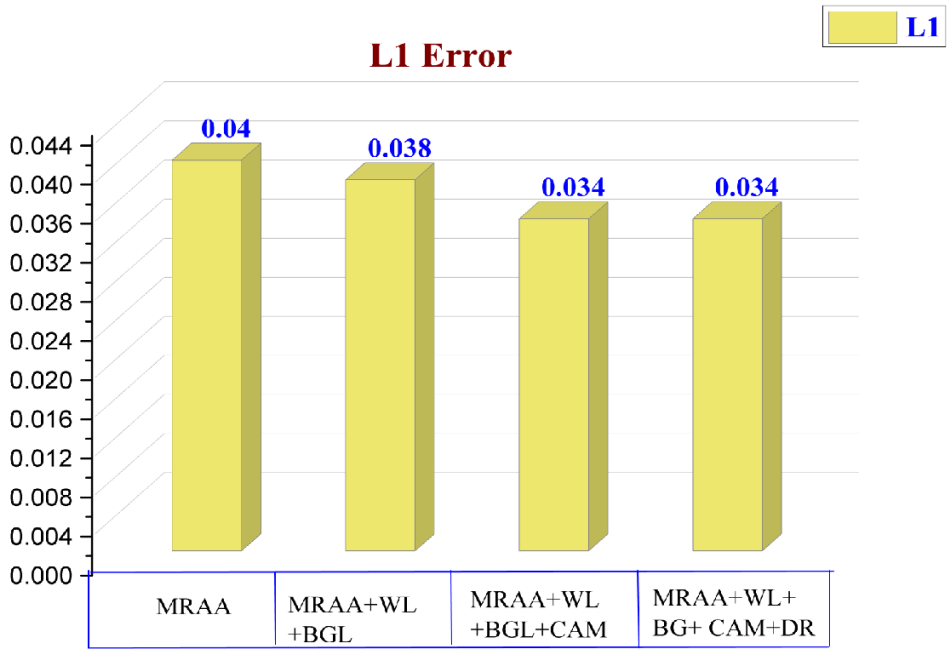


Figure 6. 11: L1 Score of the for all experiments

The y-axis indicates the L1 score and the x-axis indicates the four experiments

6.5.2. Discussion on Average Keypoint Distance Results

To assess the model's performance concerning the mean distance between the associated keypoints, we employed the average keypoint distance (AKD) method, as illustrated in section 6.2.4. As proved in Table 6.2 the MRAA scores the higher AKD which is 1.28. The MRAA+WL +BGL scores 1.21, indicating that the quality of the produced image sequences is higher than the baseline experiment. MRAA+WL+BGL+CAM score 1.17, which is a bit higher than previous experiment since they generate image sequences somehow sharp than the baseline. The MRAA+WL+BGL+CAM+DR model scores 1.13 which is the lowest of all the other experiments, suggesting that the image sequences produced by this model exhibit superior quality compared to earlier experiments.

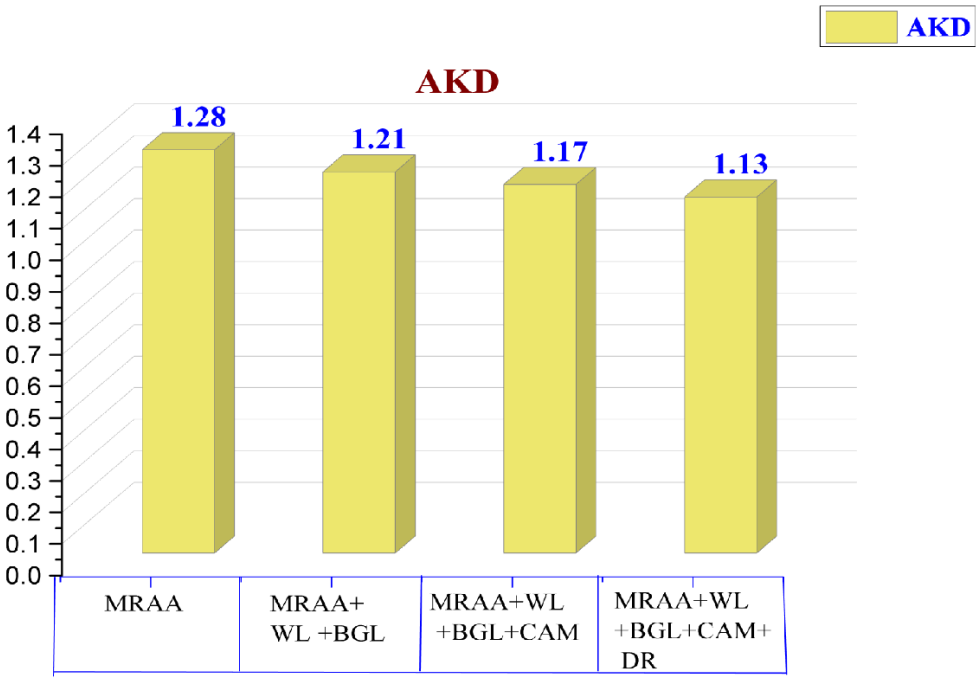


Figure 6. 12: AKD score of the for all experiments

The y-axis indicates the AKD score and the x-axis indicates the four experiments

6.5.3. Discussion on Average Euclidean Distance Results

The Average Euclidean Distance (AED) measures the degree to which the reconstructed video maintains the identity of the source image. Preserving the identity of the source image is a fundamental necessity in the context of generating talking heads. As demonstrated in Table 6.3 the MRAA scores the AED which is 0.133. The MRAA+WL+BGL scores 0.125, indicating that the quality of the produced image is higher than the prior experiment. MRAA+WL+BGL+CAM score 0.119 significantly enhance the quality of generating micro-movements related to facial expressions in human faces. MRAA+WL+BGL+CAM+DR model scores 0.115 which is the lowest of all the other experiments, suggesting that the image sequences produced by this model surpass the quality of previous experiments.

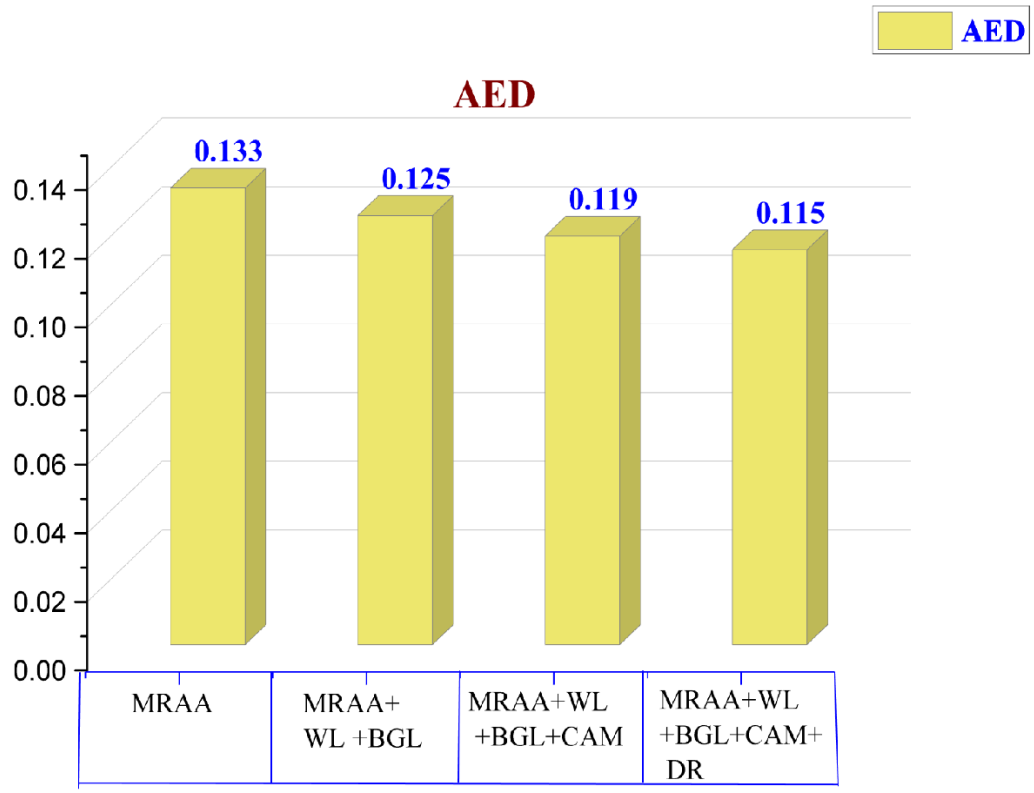


Figure 6. 13: AED score of the for all experiments

The y-axis indicates the AED score and the x-axis indicates the four experiments

6.5.4. Research Question Discussion

Every research question posed in the first chapter is addressed by this study. The following is how this study responds to each question:

Answer for Q1: Cross-modal attention models that handle two modalities with different characteristics in one model require a process of fusing two modality embeddings. Ever since the attention technique, which allows models to focus on the important parts and derive information on their own, has been devised, these studies have focused on mixing information and deriving performance using cross-modal attention for different modalities. We present a cross-modal attention mechanism that operates between keypoint maps and images, resulting in improved preservation of facial structures and the generation of micro facial movements associated with expressions. By incorporating this mechanism across multiple layers, the model

effectively captures facial geometric details across various appearance feature levels, leading to overall improved generation performance. Consequently, our model can faithfully represent facial structures at each layer by utilizing geometric information, yielding realistic human face generation. Next, the objective of this mechanism is to assist in generating motion fields used for warping the source image representations. The motion field can sometimes be contaminated with noise from the complex background, making it less capable of accurately capturing the micro-movements associated with facial expressions, as these movements originate from sparse facial keypoints. The intention here is to improve the learning of the facial motion field by reducing the impact of background noise and enhancing the representation of expression-related micro-movements.

Answer for Q2: Warp and background loss was added to the image generator model to saw the effect of this loss, the experimental result shows a better result regarding motivating the network to produce images of high quality. Based on the evaluation metrices as shown in table 6.1, 6.2 and 6.3 the warp and background loss improve the L1, AKD, and AED result from 0.040 to 0.038, 1.28 to 1.21 and 0.133 to 0.125. From this answer, this study generalized that the identified loss shows, warp and background have a positive impact to encourage the network to provide more accurate estimates of optical flow, making the warped feature maps that closely align with the feature domain of the driving video. This approach has been empirically validated and shown to be effective.

Answer for Q3: To enhance the generation of talking heads and avoid any unintended transfer of appearance or shape characteristics from the driving object, we introduced an additional disentangled representation. The experimental findings indicate an improvement in the quality of talking head images. However, it's important to note that the pose still needs to be derived from the pose branch due to the shape branch having a different pose. The model aims to animate the motion in the driving frame while maintaining the shape or identity of the source frame by disentangle shape and pose within the motion representations. Based on the evaluation metrices as shown in table 6.2 and 6.3, disentangling shape and pose clearly improve the AKD and AED result from 1.28 to 1.13 and 0.133 to 0.115. As animation through disentanglement is employed, it significantly improves the preservation of the source object's shape, resulting in a reconstructed video that exhibits enhanced temporal coherence.

CHAPTER SEVEN

7. CONCLUSION AND FUTURE WORKS

7.1. Conclusion

In recent times, there has been notable advancement in image animation tasks, thanks to the development of Generative Adversarial Networks (GANs). One noteworthy application of this progress is in the field of neural talking head synthesis.Recent research has demonstrated impressive achievements in generating realistic talking head videos through the utilization of generative adversarial networks (GANs). This thesis presents an enhanced model for creating realistic talking head videos driven by motion. It automates the generation process by blending appearance details from a source image of one individual's face with motion patterns extracted from a driving video featuring the facial expressions or body movements of another person. One of the fundamental difficulties in achieving convincing talking head simulations is the issue of imprecise reconstruction and the production of low-quality visual effects in video frame generation.

The objective of this research was to enhance the visual quality of realistic neural talking heads by adding learning constraints to the GAN network architecture trained on the most difficult dataset, known as theVoxceleb1 dataset. To do so, this thesis work adopts an efficient cross-modal attention block and adding appropriate background and warp loss along with other learning constraints that help to improve the performance of the neural network. Additionally, the model was trained by incorporating disentangled representations to produce frames capable of preserving accurate and comprehensive visual information, particularly when dealing with substantial pose changes in the animated object, while also improving visual quality.

Different experiments were performed for comparison. In the first experiment, the background loss in combination with warp loss is employed in to the image generator. It shows some improvement over the baseline work in terms of image quality and identity consistency. Additionally, strive to improve the accuracy and consistency of the predicted parameters. This experiment shows an improvement in the quality of image a little bit as shown figure 6.3 it improves the L1, AKD, and AED result from 0.040 to 0.038, 1.28 to 1.21 and 0.133 to 0.125. In the second experiment, the cross-modal attention block in combination with background and

warp loss is introduced in to the image generator decoder while the other constraints remain the same as in the first experiment. As a result, the L1 score drops from 0.040 to 0.034, the AKD score drops from 1.28 to 1.17, and the AED score drops from 0.133 to 0.119. This addresses the first research question of what effect would using cross-modal attention block have on the quality of realistic neural talking heads. In the fourth experiment, to improve the talking head generation and avoid the leakage of the appearance or identity of the driving object disentangled representation was added. This model tried to make the image sharp and also to maintain the original identity of the source image. It improves both the AKD score from 1.28 to 1.13, and AED score from 0.133 to 0.115 but, the L1 score slightly the same with the previous experiment. This addresses the third research question.

In this study, we introduce a multi-occlusion network designed to enhance image details across multiple scales, aiming for greater accuracy in our results. We leverage the occlusion map's capability to rectify pixel values in the generated images, ultimately leading to improved visual outcomes. We apply this approach to resolutions of (64, 64), (128, 128), and (256, 256), allowing us to refine the network's feature representation at various resolutions and boost its image repair capabilities. Furthermore, it's worth noting that our model can be effectively trained in an unsupervised fashion.

Finally, this study finds that using the background and warp loss in the generator network and cross-modal attention module in the generator decoder improves the quality of talking head generation by effectively preserving structural and textural information while significantly reducing noise and artifacts. Secondly, we employ multi-resolution occlusion masks to enhance feature fusion more effectively, in contrast to relying on a single occlusion mask. Finally, it is reasonable to conclude that adding disentangled representation to improve image quality, obtain a better temporal consistency and preserve identity-related information. The experiments conducted on the Voxceleb1 dataset showcased that MRAA+WL+BGL+CAM+DR outperforms state-of-the-art methods in preserving identity, image quality, reconstruction quality, and animation quality.

7.2. Future Work

While our proposed model has indeed brought about significant enhancements in the synthesis of realistic neural talking heads, it is important to acknowledge that there remain certain

constraints and limitations. This work attempts to fill one of the many gaps in the present talking head synthesis by adopting cross-modal attention mechanism and use additional auxiliary loss functions motivating the network to produce images of higher quality. This thesis approach does have certain restrictions, though. Due to time constraints, the proposed model could only be trained only on Voxceleb1 datasets that were publicly available; however, in order for it to be suitable for addressing this limitation the proposed model, it must also be trained on different datasets in an unsupervised manner. Like full human body and publicly available high-resolution dataset.

Another limitation of this study is the lack of consideration for inter-frame correlation. In our future work, we intend to integrate neural networks, like LSTM, to store the generated results from the previous frame. This stored information can then be utilized to enhance the generation of the current frame. Given the inherent connection between consecutive frames, some content generated from the previous frame can contribute to improving the quality of the current frame reconstruction. Additionally, we plan to explore the use of multiple source frame images captured from various angles to construct potential source frames. We will automate the adjustment of the contribution of source frames from different angles using neural networks, with the aim of generating more precise and realistic reconstructed frames in our future work.

* * *

Special Acknowledgment

This research work was funded by grant number ASTU/SM-R/808/23 from Adama Science and Technology University, **Adama, Ethiopia**.

References

Amos, B., Ludwiczuk, B., & Satyanarayanan, M. (2016). *OpenFace : A general-purpose face recognition library with mobile applications*. *June*.

Bengio, Y., & Haffner, P. (1998). *Gradient-Based Learning Applied to Document Recognition*. *86*(11).

Bulat, A., & Tzimiropoulos, G. (2017). How Far are We from Solving the 2D & 3D Face Alignment Problem? (and a Dataset of 230,000 3D Facial Landmarks). *Proceedings of the IEEE International Conference on Computer Vision*, *2017-Octob*, 1021–1030. https://doi.org/10.1109/ICCV.2017.116

Burkov, E., Pasechnik, I., … A. G.-P. of the, & 2020, U. (2020). Neural head reenactment with latent pose descriptors. *Openaccess.Thecvf.Com*. http://openaccess.thecvf.com/content_CVPR_2020/html/Burkov_Neural_Head_Reenactment_with_Latent_Pose_Descriptors_CVPR_2020_paper.html

Demir, U., & Unal, G. (2018). *Patch-Based Image Inpainting with Generative Adversarial Networks*. http://arxiv.org/abs/1803.07422

Goodfellow, I. J., Pouget-abadie, J., Mirza, M., Xu, B., & Warde-farley, D. (2014). *Generative Adversarial Nets*. 1–9.

Ha, S., Kersner, M., Kim, B., Seo, S., & Kim, D. (2020). MarioNETte: Few-shot face reenactment preserving identity of unseen targets. *AAAI 2020 - 34th AAAI Conference on Artificial Intelligence*, 10893–10900. https://doi.org/10.1609/aaai.v34i07.6721

Hao, H., Baireddy, S., Reibman, A. R., & Delp, E. J. (2020). *FaR-GAN for One-Shot Face Reenactment*.

He, K., Zhang, X., Ren, S., Recognition, J. S. pattern, & 2016, U. (2016). Deep residual learning for image recognition. *Openaccess.Thecvf.Com*. http://openaccess.thecvf.com/content_cvpr_2016/html/He_Deep_Residual_Learning_CVPR_2016_paper.html

Hong, F. T., Zhang, L., Shen, L., & Xu, D. (2022). Depth-Aware Generative Adversarial

Network for Talking Head Video Generation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *2022-June*(8), 3387–3396. https://doi.org/10.1109/CVPR52688.2022.00339

Huang, X., Liu, M. Y., Belongie, S., & Kautz, J. (2018). Multimodal Unsupervised Image-to-Image Translation. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *11207 LNCS*, 179–196. https://doi.org/10.1007/978-3-030-01219-9_11

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

Johnson, J., Alahi, A., & Fei-Fei, L. (2016). Perceptual losses for real-time style transfer and super-resolution. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *9906 LNCS*, 694–711. https://doi.org/10.1007/978-3-319-46475-6_43

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.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, *60*(6), 84–90. https://doi.org/10.1145/3065386

Liu, J., Wang, X., Fu, X., Chai, Y., Yu, C., Dai, J., & Han, J. (2023). *OPT: One-shot Pose-Controllable Talking Head Generation*. *2020*, 1–5. https://doi.org/10.1109/icassp49357.2023.10094598

Liu, M. Y., Huang, X., Mallya, A., Karras, T., Aila, T., Lehtinen, J., & Kautz, J. (2019). Few-shot unsupervised image-to-image translation. *Proceedings of the IEEE International Conference on Computer Vision*, *2019-Octob*, 10550–10559. https://doi.org/10.1109/ICCV.2019.01065

Lorenz, D., Bereska, L., Milbich, T., & Ommer, B. (2019). Unsupervised part-based disentangling of object shape and appearance. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *2019-June*, 10947–10956. https://doi.org/10.1109/CVPR.2019.01121

Meshry, M., Suri, S., … L. D.-P. of the, & 2021, U. (2021). Learned Spatial Representations for Few-shot Talking-Head Synthesis. *Openaccess.Thecvf.Com*. http://openaccess.thecvf.com/content/ICCV2021/html/Meshry_Learned_Spatial_Representations_for_Few-Shot_Talking-Head_Synthesis_ICCV_2021_paper.html

Mirza, M., & Osindero, S. (2014). *Conditional Generative Adversarial Nets*. https://arxiv.org/abs/1411.1784v1

Nagraniy, A., Chungy, J. S., & Zisserman, A. (2017). VoxCeleb: A large-scale speaker identification dataset. *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, *2017-Augus*, 2616–2620. https://doi.org/10.21437/Interspeech.2017-950

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

Oh, J., Guo, X., Lee, H., Lewis, R., & Singh, S. (2015). Action-conditional video prediction using deep networks in Atari games. *Advances in Neural Information Processing Systems*, *2015-Janua*, 2863–2871.

Rehusevych, O., & Kupyn, O. (2020). *One-Shot Identity Preserving Photorealistic Face Reenactment*.

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

Science, C. (2022). *ep rin t n ot pe er r ev ed Pr ep rin t n ot pe er r ed*.

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).

Siarohin, A., Sangineto, E., Sebe, N., & Rhone-alpes, I. G. (n.d.). *Deformable GANs for Pose-based Human Image Generation*.

Siarohin, A., Woodford, O. J., Ren, J., Chai, M., & Tulyakov, S. (2021). Motion representations for articulated animation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 13648–13657. https://doi.org/10.1109/CVPR46437.2021.01344

Srivastava, N., Mansimov, E., & Salakhutdinov, R. (2015). Unsupervised learning of video representations using LSTMs. *32nd International Conference on Machine Learning, ICML 2015*, *1*, 843–852.

Tao, J., Wang, B., Ge, T., Jiang, Y., Li, W., & Duan, L. (2022). *Motion Transformer for Unsupervised Image Animation*.

Tran, L., Yin, X., & Liu, X. (2017). Disentangled representation learning GAN for pose-invariant face recognition. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, *2017-Janua*, 1283–1292. https://doi.org/10.1109/CVPR.2017.141

Tripathy, S., Kannala, J., & Rahtu, E. (2021). FACEGAN: Facial attribute controllable rEenactment GAN. *Proceedings - 2021 IEEE Winter Conference on Applications of Computer Vision, WACV 2021*, 1328–1337. https://doi.org/10.1109/WACV48630.2021.00137

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

Ulyanov, D., Vedaldi, A., & Lempitsky, V. (2016). *Instance Normalization: The Missing Ingredient for Fast Stylization*. *2016*. http://arxiv.org/abs/1607.08022

VanderPlas, J. (2016). *Python data science handbook: Essential tools for working with data*. https://books.google.com/books?hl=en&lr=&id=xYmNDQAAQBAJ&oi=fnd&pg=PR2&dq=Python+Data+Science+Handbook&ots=XrePq0rn7L&sig=iEqGd4vQfnMQ9EmKsjQqMQIjEk8

Vetter, T. (1999). *A Morphable Model For The Synthesis Of 3D Faces f g*. 187–194.

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.

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., Tao, A., Liu, G., Kautz, J., & Catanzaro, B. (2019). *Few-shot Video-to-Video Synthesis*.

Wang, W., Alameda-Pineda, X., Xu, D., Fua, P., Ricci, E., & Sebe, N. (2018). Every Smile is Unique: Landmark-Guided Diverse Smile Generation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 7083–7092. https://doi.org/10.1109/CVPR.2018.00740

Wiles, O., Koepke, A. S., & Zisserman, A. (2018). X2Face: A Network for controlling face generation using images, audio, and pose codes. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *11217 LNCS*, 690–706. https://doi.org/10.1007/978-3-030-01261-8_41

Zakharov, E., Shysheya, A., Burkov, E., & Lempitsky, V. (2019). Few-shot adversarial

learning of realistic neural talking head models. *Proceedings of the IEEE International Conference on Computer Vision*, *2019-Octob*, 9458–9467. https://doi.org/10.1109/ICCV.2019.00955

Zhao, J., & Zhang, H. (2022). *Thin-Plate Spline Motion Model for Image Animation*. 3657–3666.

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

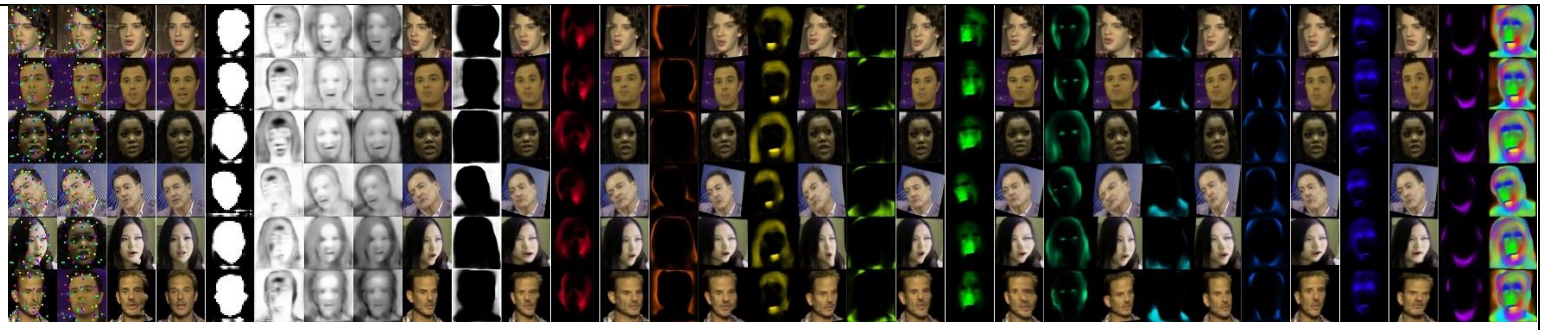
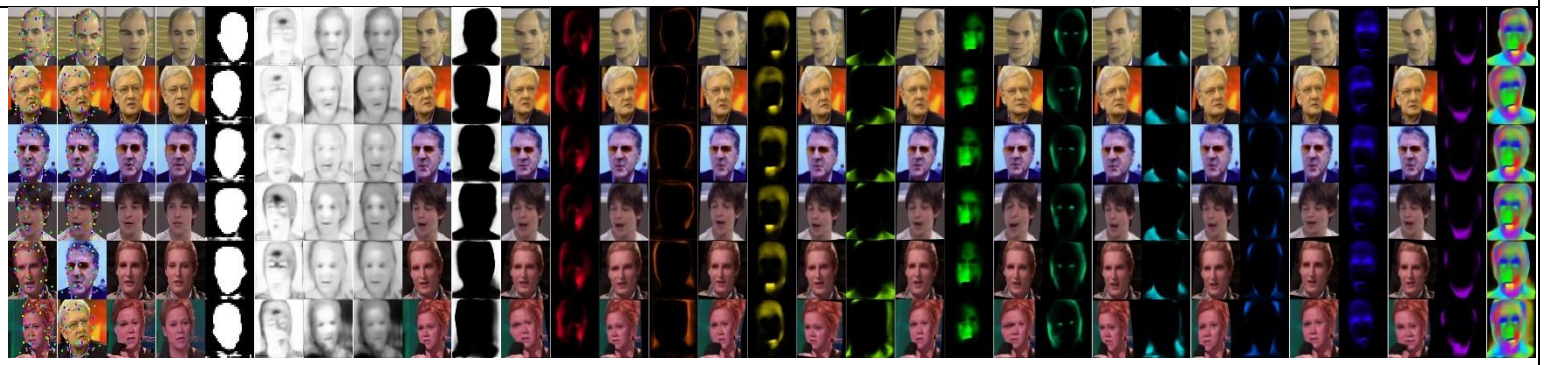
APPENDICES

Appendix A: Ablation Study

We conducted ablation experiments using the Voxceleb1 dataset to demonstrate the enhancements brought about by each of our proposed components. Furthermore, we carried out an ablation study to comprehensively assess our method, examining the significance of individual components and identifying the combinations that result in high-quality images. To begin with, we substituted the keypoint detector with a region predictor, as used in the baseline approach, which predicts a single occlusion map. It is important to highlight that introducing the region predictor-based representation by itself had only a slight effect on the L1 score, primarily influenced by the considerably larger background region. However, it had a more pronounced impact on other metrics, particularly those that are more responsive to errors related to object parts within facial objects. Nevertheless, it had a more noticeable influence on other metrics, specifically those that are more sensitive to inaccuracies related to facial object components. Furthermore, we noticed that when our keypoint detector-based approach is omitted, the network inadvertently incorporates some of the motion information into the background branch. This, in turn, results in a notable deterioration in the quality of keypoints, subsequently affecting the AKD and AED scores adversely.

Ablation study for video reconstruction by replacing keypoint detector with region-based predictor with only estimate single occlusion map.

Appendix B: Result on different epoch

Epoch 1	
Epoch 50	
Epoch 100	

Appendix C: List of Sample Code

Identity Encoder Network Implementation:

```
class AVDNetwork(nn.Module):
    def __init__(self, num_tps, id_bottle_size=64, pose_bottle_size=64):
        super(AVDNetwork, self).__init__()
        input_size = 5*2 * num_tps
        self.num_tps = num_tps
        self.id_encoder = nn.Sequential(
            nn.Linear(input_size, 256),
            nn.BatchNorm1d(256),
            nn.ReLU(inplace=True),
            nn.Linear(256, 512),
            nn.BatchNorm1d(512),
            nn.ReLU(inplace=True),
            nn.Linear(512, 1024),
            nn.BatchNorm1d(1024),
            nn.ReLU(inplace=True),
            nn.Linear(1024, id_bottle_size)
        )
```

Pose Encoder Network Implementation:

```
class AVDNetwork(nn.Module):
    def __init__(self, num_tps, id_bottle_size=64, pose_bottle_size=64):
        super(AVDNetwork, self).__init__()
        input_size = 5*2 * num_tps
        self.num_tps = num_tps

        self.pose_encoder = nn.Sequential(
            nn.Linear(input_size, 256),
            nn.BatchNorm1d(256),
            nn.ReLU(inplace=True),
            nn.Linear(256, 512),
            nn.BatchNorm1d(512),
            nn.ReLU(inplace=True),
            nn.Linear(512, 1024),
            nn.BatchNorm1d(1024),
            nn.ReLU(inplace=True),
            nn.Linear(1024, pose_bottle_size)
        )
```

Decoder Network Implementation:

```
class AVDNetwork(nn.Module):
    def __init__(self, num_tps, id_bottle_size=64, pose_bottle_size=64):
        super(AVDNetwork, self).__init__()
        input_size = 5*2 * num_tps
        self.num_tps = num_tps
        self.decoder = nn.Sequential(
            nn.Linear(pose_bottle_size + id_bottle_size, 1024),
            nn.BatchNorm1d(1024),
            nn.ReLU(),
            nn.Linear(1024, 512),
            nn.BatchNorm1d(512),
            nn.ReLU(),
            nn.Linear(512, 256),
            nn.BatchNorm1d(256),
            nn.ReLU(),
            nn.Linear(256, input_size)
        )
    def forward(self, kp_source, kp_random):
        bs = kp_source['fg_kp'].shape[0]
        pose_emb = self.pose_encoder(kp_random['fg_kp'].view(bs, -1))
        id_emb = self.id_encoder(kp_source['fg_kp'].view(bs, -1))
        rec = self.decoder(torch.cat([pose_emb, id_emb], dim=1))
        rec =  {'fg_kp': rec.view(bs, self.num_tps*5, -1)}
        return rec
```

Discriminator Network Implementation

```
class DiscriminatorFullModel(torch.nn.Module):
    def __init__(self, kp_extractor, generator, discriminator, train_params):
        super(DiscriminatorFullModel, self).__init__()
        self.kp_extractor = kp_extractor
        self.generator = generator
        self.discriminator = discriminator
        self.train_params = train_params
        self.scales = self.discriminator.scales
        self.pyramid = ImagePyramide(self.scales, generator.num_channels)
        if torch.cuda.is_available():
            self.pyramid = self.pyramid.cuda()
        self.loss_weights = train_params['loss_weights']
```

```
   def forward(self, x, generated):
      pyramide_real = self.pyramid(x['driving'])
      pyramide_generated = self.pyramid(generated['prediction'].detach())
      kp_driving = generated['kp_driving']
      discriminator_maps_generated = self.discriminator(pyramide_generated,
kp=detach_kp(kp_driving))
      discriminator_maps_real = self.discriminator(pyramide_real, kp=detach_kp(kp_driving))
      loss_values = {}
      value_total = 0
      for scale in self.scales:
         key = 'prediction_map_%s' % scale
         value = (1 - discriminator_maps_real[key]) ** 2 + discriminator_maps_generated[key]
** 2
         value_total += self.loss_weights['discriminator_gan'] * value.mean()
      loss_values['disc_gan'] = value_total
      return loss_values
```