Investigative Study on Generative Models for Face2Sketch

Name : Yi-Chun Huang, Student ID : 36228559

***Abstract*—Face2Sketch generation is an important task in computer vision and has wide applications in digital entertainment, forensic art, and facial recognition systems. In this paper, we propose a novel model, Convolutional Autoencoder with Self-Attention and Quantization (CASQ), for generating high-quality sketches from facial images. CASQ utilizes a combination of self-attention and quantization techniques to capture the complex features of faces and generate detailed and accurate sketches. We compare the performance of CASQ with two other models, deep autoencoder and variational autoencoder, on CUHK face sketch dataset. Our results demonstrate that CASQ outperforms the other models in terms of both visual quality and quantitative evaluation metrics, achieving state-of-the-art performance on the dataset.**

***Index Terms*—Face2Sketch, convolutional autoencoder, deep autoencoder, variational autoencoder, self-attention, quantization.**

I. Introduction

The first paper on face recognition was "Pattern Recognition and Reading by Machine", published by American scholar Woodrow Wilson Bledsoe in 1964 [1]. The paper proposed a linear algebra-based method for recognising faces on black and white photographs. Bledsoe's paper opened the way for research in the field of face recognition and laid the foundation for later face recognition techniques. The baton was picked up in the 1970s by "Identification of Human Faces" [2], and although accuracy improved, measurement and position still required manual calculation, which proved to be extremely labour intensive. It was not until the late 1980s that breakthroughs were made in facial recognition, with Sirovich and Kirby [3] proposing a method based on Principal Component Analysis (PCA) and Belhumeur et al [4] proposing a method based on Linear Discriminant Analysis (LDA). Since then, facial recognition technologies are emerging rapidly like mushrooms after rain.

In the past few decades, there has been an increasing interest in face recognition theories and algorithms [5]. This technology is known to be used in areas such as security, identity registration, image retrieval, and human-computer interaction [6]. There are several extensions to face recognition technology, one of which is "Face2Sketch", a technology that automatically generates a sketch or drawing of a face through computer algorithms and artificial intelligence techniques to identify a potential suspect or witness based on the facial description provided by the suspect or witness [7]. The process of face sketch recognition involves the use of computer algorithms and artificial intelligence techniques to analyse facial features, such as the shape of the eyes, nose and mouth, and the contours of the face, and then generate a face sketch or drawing based on these features that resembles the person being described. In addition to its use in crime investigations, face sketch recognition can be applied to biometric systems, public safety and surveillance, entertainment and creative arts [8]. It can improve the efficiency and accuracy of investigations, and enhance the security of various systems and environments.

Areas of application for Face2Sketch include the following:

- Crime detection: Face2Sketch can help the police to quickly generate sketches of similar facial features of suspects, improving the efficiency of crime detection.
- Judicial identification: In judicial cases, Face2Sketch can be used to identify suspects or witnesses, helping investigators to gather more evidence.
- Security monitoring: Face2Sketch can be used for security monitoring in public places such as airports, train stations, shopping malls, schools, etc. It can quickly identify suspected persons and enhance the security of the premises.
- Biometric identification: Face2Sketch can be applied to biometric systems such as access control systems, time and attendance systems, payment systems, etc. to improve the security and accuracy of the system.
- Entertainment and culture: Face2Sketch technology can be applied to entertainment and culture, such as face sketching in cartoon characters, comics, games, etc., to help creators quickly generate sketches of characters' facial features.

Through Face2Sketch, we can increase the efficiency of crime detection, improve justice identification, enhance security in public places, improve system security, and support creativity and entertainment. The importance of Face2Sketch continues to spread and grow in a variety of areas. Despite the potential risks and challenges associated with face recognition technology, such as privacy protection and false positive rates, it is still a very useful technology that can bring many benefits to society.

In summary, the contributions of this paper are as follows:

1) Describing real-world applications of Face2Sketch and the benefits and drawbacks
2) Comment on some existing recent work and approaches for Face2Sketch
3) *Contribution*: this paper presents a novel application based on convolutional autoencoder - convolutional autoencoder with self-attention and quantization (CASQ)
4) Comparison and analysis of some existing models with CASQ applied to CUHK face sketch database (CUFS)

[9]

5) Highlighting possible breakthrough directions and subsequent extensions of CASQ

II. Literature Review

In recent years, Face2Sketch conversion has become an increasingly popular research topic in the field of computer vision and machine learning [10]. The primary objective of Face2Sketch conversion is to generate sketches that can capture the essential characteristics of the input face image. Several approaches have been proposed to achieve this goal, including traditional computer vision techniques, as well as more advanced deep learning-based methods.

A. Autoencoder

Autoencoder (AE) is an unsupervised learning algorithm for multi-layer neural networks that helps classify, visualise and store data. The architecture can be subdivided into two parts, encoder and decoder, which perform compression and decompression actions respectively, allowing the output and input values to represent the same meaning [11].

By reconstructing the neural network training process of the input, the vectors of the hidden layer have a dimensionality reduction effect [12]. The special feature is that the encoder creates a hidden layer (or layers) containing a low-dimensional vector of the input information. A decoder then reconstructs the input data from the low-dimensional vectors in the hidden layer. The final AE is trained by the neural network to obtain a low-dimensional vector representing the input data in the hidden layer. Figure 1 presents the model architecture of the autoencoder.

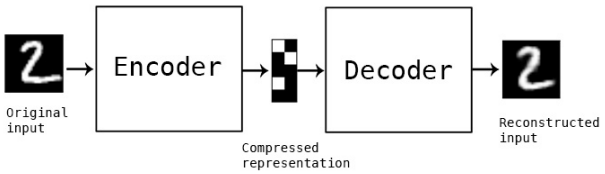

Fig. 1: AE model architecture[1]

B. Generative Adversarial Networks

Generative adversarial networks (GANs) are neural networks made up of two CNNs, with two components, the generator and the discriminator. GANs are like a question-and-answer system, with the generator constantly asking questions and the discriminator answering them. In [13], an analogy was drawn between the relationship between police and criminals. In the story, the criminals are constantly creating fake money to deceive the eyes of the police, and every time they are caught, they retool their techniques to create fake money. Figure 2 shows how GANs works.

[1] https://blog.keras.io/
[2] https://sthalles.github.io/

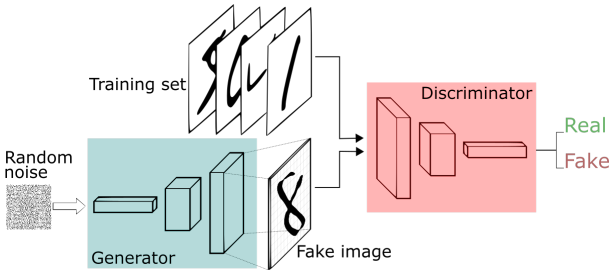

Fig. 2: How GANs works[2]

C. Recent Works and Methods

Most of the recent research on Face2Sketch is based on the section II-A II-B.

Bi et al [10] propose to learn mapping by cGAN and focus multi-scale images to obtain image information at different resolutions. Tang et al [14] proposed a novel deep learning model (ASGAN), which consists of two pairs of generators and discriminators, one pair generates faces with attributes, and the other pair is used to generate faces with attributes for image to Translation of the sketches, forming a W-net. Qi et al [15] proposed a novel semantic-driven generative adversarial network (SDGAN) and a novel class-level knowledge adaptive reweighting loss (ARLoss) to balance the importance of different semantic parts. Hou et al [16] proposed to introduce a new cross-domain latent modulation mechanism into the variational autoencoder framework for efficient transfer learning.

D. Limitations and Motivation

AEs and GANs are very similar in generating images. AEs generate new data by encoder, while GANs generate random images by generator. The difference is that AE optimizes the hidden variables by comparing the difference between the generated data and the original data; GANs compare the difference between the real data and the generated data by the discriminator, so that the two images can produce similar results.

Since AE's hidden variables are generated from the input data encoder, the hidden variables correspond more to the original data and the generated data are more regular. However, this results in the data being supervised point-to-point, which means that every point is guaranteed to be the same, and this results in the global information not being obtained comprehensively enough, with weak global information and strong local information [17] [18].

The disadvantage of GANs is that the input and data correspondence is weak, which makes the generated data easily deviate from the topic, and the discriminator is judging the data as a whole, so the generated data is more continuous and can have a better grasp of the data as a whole, that is, the global information is strong and the local information is weak [17] [18].

This paper is to solve the deficiency of traditional AE for global information. Therefore, the self-attention mechanism and quantization are introduced to enhance the ability of the convolutional autoencoder to obtain global information.

III. PRELIMINARY ON METHODOLOGIES

In this paper, Deep Autoencoder (DAE) and Variational Autoencoder (VAE) are compared with the proposed CASQ, and structural similarity index measure (SSIM) and mean squared error (MSE) are used for evaluation.

A. DAE

DAE is a neural network structure consisting of several hidden layers designed to convert high-dimensional data into a low-dimensional representation while preserving the important features of the original data [19]. It can be trained by back propagation algorithms to minimise reconstruction errors so that the output is as close to the input as possible. The architecture of the DAE consists of an encoder and a decoder, as shown in the figure 3.

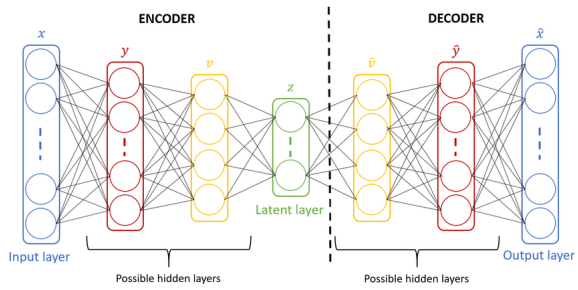

Fig. 3: DAE architecture[3]

The encoder consists of a number of hidden layers used to compress the input data into a low-dimensional representation. The decoder is also composed of several hidden layers and is used to reconstruct the low-dimensional representation back into the original input data. The training process of the DAE can be achieved by the following steps:

1) Feeding the input data into the encoder to obtain a low-dimensional representation.
2) Feed the low-dimensional representation into the decoder to obtain the reconstructed data.
3) The reconstruction error is calculated and the weights of the encoder and decoder are updated by a back propagation algorithm.

The advantages of deep auto-encoders include the ability to learn non-linear features and the ability to apply them to different types of data such as images, sounds and text [20]. However, it has the disadvantage of being prone to overfitting and is sensitive to the choice of hyperparameters such as initial parameters and the number of hidden layers.

B. VAE

VAE is a deep learning-based generative model for learning low-dimensional representations of data and generating new data. Unlike traditional autoencoders, the VAE uses a Latent Variable model to perform low-dimensional representations in a more logical way. It can be trained by maximising the lower bound on the variables and converting the distribution of the latent variables to a standard normal distribution by means of a re-parameterisation technique, allowing the latent variables to be randomly sampled and new data to be generated. The architecture of the VAE consists of an encoder and a decoder, as shown in the figure 4.

Similar to the DAE, the encoder consists of a number of hidden layers used to convert input data into distribution parameters of potential variables. The VAE training process can be achieved by the following steps.

1) The input data is fed into the encoder to obtain the distribution parameters of the potential variable.
2) Sampling from the distribution of potential variables to obtain a random potential variable.
3) Feed the random potential variable into the decoder to obtain the reconstructed data.
4) The reconstruction error and Kullback-Leibler divergence are calculated and the weights of the encoder and decoder are updated by a back propagation algorithm.

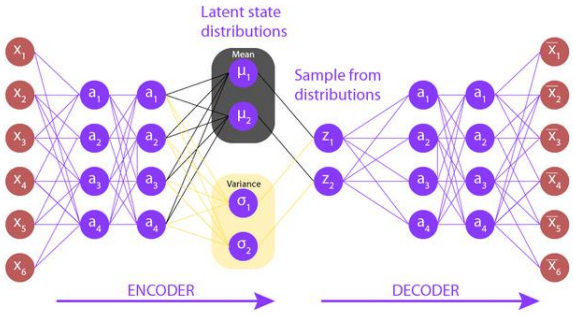

Fig. 4: VAE architecture[4]

The advantage of VAE is that new data can be generated and the characteristics of the generated data can be controlled by the distribution parameters of the potential variables. The disadvantage is that it is computationally complex and still has difficulties in generating high quality data [21].

C. CASQ

CASQ is a convolutional neural network (CNN)-based autoencoder whose main feature is the use of self-attention mechanisms and quantization techniques to effectively reduce the computational and storage costs of the model while maintaining a high quality of reconstructed data.

CASQ consists of three main components: Convolutional Autoencoder, Self-Attention and Quantization. Of these three components, Convolutional Autoencoder is responsible for feature extraction and reconstruction, Self-Attention is used to further Self-Attention is used to further improve feature extraction, and Quantization is used to reduce storage and computational costs.

1) Convolutional Autoencoder:

Convolutional Autoencoder (CAE) is an autoencoder that uses a Convolutional Neural Network (CNN) for image feature extraction and restoration.The goal of CAE is to learn a

[3] https://www.researchgate.net/

[4] https://www.geeksforgeeks.org/

compressed representation that compresses the original image to a smaller dimension while preserving the important features of the image, and then restores the compressed representation to the original image through a decoder. CAE can be applied to image processing tasks such as image compression, image noise reduction and image deblurring.

The CAE architecture consists of two parts: the encoder and the decoder. The encoder takes the original image through a series of convolutional and pooling layers to extract and compress the features to produce a smaller dimensional feature vector. The decoder decodes and restores the compressed representation through a series of deconvolutional and upsampling layers, resulting in the same image as the original. The encoder and decoder are both composed of a CNN, where the convolutional and deconvolutional layers are used to extract and restore image features, and the pooling and upsampling layers are used to reduce the dimensionality of the feature map [22].

The figure 5 shows a schematic representation of the architecture of the CAE encoder and decoder.

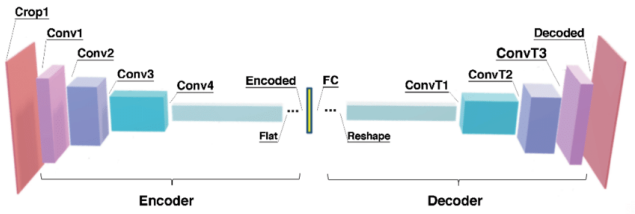

Fig. 5: CAE architecture[5]

a) Convolutional Layer: The convolutional layer is one of the core components of a convolutional neural network. It performs convolutional operations on the input data by applying a small matrix called a convolutional kernel to extract the local features of the data. The convolutional layer effectively reduces the number of parameters and makes the neural network translation invariant (i.e. translation invariant to the input data).

b) Pooling Layer: The pooling layer is another important component of a convolutional neural network. It reduces the spatial size of the input data by performing downsampling operations on the output of the convolutional layer. Pooling can effectively reduce the computational effort and enhance the robustness of the neural network to translation invariance. Common pooling operations include maximum pooling and average pooling.

c) Deconvolutional Layer: In CAE, deconvolutional layers are commonly used to restore encoded low-dimensional feature vectors to their original high-dimensional images. These layers can be seen as the inverse of forward convolution, which uses transpose convolution to restore the original spatial dimensions.

Specifically, deconvolutional layers typically consist of three steps:

1) linear transformation of the encoded low-dimensional feature vector to extend the size of the feature map.
2) A forward convolution of the expanded feature map to further restore the spatial dimensions of the feature map.
3) During the convolution operation, the padding and stride parameters are used to control the size and shape of the image.

d) Upsampling Layer: In addition to deconvolution using deconvolutional layers, upsampling layers are also a common technique used to restore the size of a feature map to its original size. Upsampling layers typically use interpolation methods (e.g. nearest neighbour interpolation, bilinear interpolation, etc.) to scale low resolution feature maps to a higher resolution. They can increase the size of the feature map by a factor of one in order to better restore the original image. Often, the parameters of the upsampling layers (e.g. stride, kernel size, etc.) can be adjusted to change the extent and method of upsampling.

The CAE training process can be implemented using the Backpropagation algorithm. First, the original image is fed into an encoder, which converts the image into a compressed representation. The compressed representation is then fed into the decoder, which restores the compressed representation to the original image. Next, the reconstruction error between the restored image and the original image is calculated and the parameters of the encoder and decoder are updated using a back propagation algorithm. This process is repeated until the model converges or the maximum number of training sessions is reached.

2) Self-Attention:

Self-attention is a mechanism for taking into account the relationships between different locations. It was first widely used in the field of natural language processing and has also been successful in the field of images.

In the image domain, the main role of self-attention is to model the interdependencies between different locations in an image. For instance, in a photograph of a person's face, the position and relationship between features such as the eyes, nose and mouth are very important, as they together determine the recognition and expression of the face. self-attention mechanisms can help models automatically learn the relationships between these interdependent features, thereby improving the accuracy and efficiency of tasks such as image classification, recognition and generation.

The principle of the self-attention mechanism is based on matrix computing, which models the interdependencies between different locations by computing an attention weight matrix. Specifically, given a feature mapping, the self-attention mechanism calculates the similarity between the feature vectors of each location and the feature vectors of other locations, and then uses these similarities to calculate the attention weights of each location to the other locations. Finally, based on these weights, the feature mappings can be weighted and averaged or linearly combined to produce a combined feature vector, thus enabling the interaction and integration of information between different locations.

The self-attention mechanism consists of three main steps:

1) Calculating Attention Weights: For each word (or other input unit), a weight is calculated that indicates how the model should pay attention to that unit in the input

[5]https://www.researchgate.net/

sequence. Attention weights are usually calculated by the following formula [23].

$$Attention(q,k,v) = softmax\left(\frac{qk^T}{\sqrt{d_k}}\right)v \quad (1)$$

where q, k and v denote the query vector, key vector and value vector respectively, and d_k is the dimension of the vector. The softmax function normalises the attention weights and multiplies them by the value vector to obtain the final attention value.

2) Apply Attention Weights: A weighted average of the value vectors at each input position is used to generate a weighted input vector using the calculated attention weight weights.
3) Multi-Head Self-Attention: In order to improve the performance of a model, a self-attention mechanism usually consists of multiple "heads", each of which can learn different attention patterns. Typically, multi-head self-attention is performed by dividing the input vector into sub-vectors, each of which is used to compute a separate attention weight. Eventually, the attention weights of all the subvectors are weighted and combined to produce the final output vector.

Self-attention mechanisms have a wide range of applications in image processing. It can help models automatically learn the interdependencies between different locations in an image, thereby improving the accuracy and efficiency of tasks such as image classification, recognition and generation. In addition, it can capture long-range correlation, better capture global information of images, reduce the number of model parameters, improve the efficiency and generalisation of models, and adapt to different scales of images. Overall, self-attention is a valuable image processing technique that can play an important role in a variety of image-related tasks.

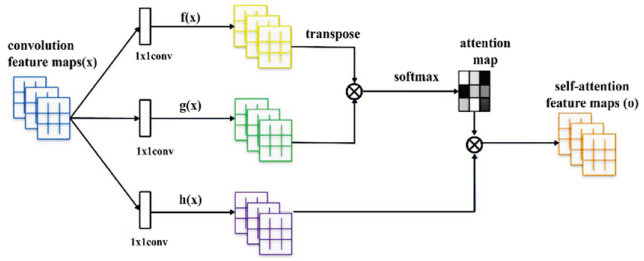

Fig. 6: Self-attention module. (Zhang et al, 2019) [24]

3) Quantization:

The Vector Quantizer (VQ) layer is a compression technique commonly used in neural networks as an unsupervised method for quantizing a continuous signal or a continuous set of feature vectors into a finite discrete codebook.The VQ layer used in this paper is referenced to Van Den Oord, A., Vinyals, O. (2017) [25] for Discrete Latent variables.

In neural networks, the VQ layer is mainly used to compress the middle layer features. It maps the continuous feature vectors into a discrete codebook, each of which consists of a number of vectors called "codebook vectors", usually learned from a large amount of training data during training through clustering algorithms such as k-means. During the compression process, each input feature vector is mapped to the closest discrete codebook vector, which is used as the output of the compression.

The following is the basic principle of the VQ layer.

- Model output: The output of the VQ layer is a matrix of size $n_{codebook}$, where each row represents a discrete codebook vector. When the input vector x is quantized into the codebook vector c_i, the corresponding output vector has the first i element as 1 and the remaining elements as 0.
- Quantization loss: The training process of the VQ layer can be thought of as a process of minimising quantization loss, where quantization loss is the distance between the input vector x and the discrete codebook vector c_i it is quantized into. Euclidean distance or Manhattan distance is usually used as the distance measure.
- Training Objective: The training objective of the VQ layer is to minimise the average quantitative loss of all training samples, i.e. to minimise the following target functions:

$$L = \frac{1}{N}\sum_{i=1}^{N} \parallel x_i - e_{q_i} \parallel_2^2 \quad (2)$$

where N is the number of training samples, x_i is the ith training sample, q_i is the discrete codebook index of x_i, and e_{q_i} is the discrete codebook vector corresponding to q_i.

- Learning rule: In the VQ layer, we need to update the discrete codebook vectors to minimise quantization loss. A common learning rule is to set the discrete codebook vectors to the closest mean value of their corresponding training samples, i.e.

$$e_j = \frac{1}{N_j}\sum_{i=1}^{N} x_i[q_i = j] \quad (3)$$

where N_j is the sample number of $q_i = j$ and $x_i[q_i = j]$ is the jth component of the sample x_i.

- Training process: The training process for the VQ layer usually consists of the following steps:
  1) Random initialization of discrete codebook vectors.
  2) For each training sample, find the nearest discrete codebook vector and quantize the sample as that vector, calculating the quantization loss.
  3) Updating discrete codebook vectors using quantization losses.
  4) Repeat steps 2-3 until the astringent condition is met.

In the Face2Sketch domain, the VQ layer can be used as part of the encoder to compress the face image into a low-dimensional discrete vector for the Face2Sketch transformation. Compared to traditional image conversion methods, the VQ layer has the following advantages and benefits in the Face2Sketch domain.

- Better image quality: The VQ layer captures complex details and features in the original image and converts

them into a high quality sketch image. the VQ layer also learns smaller discrete code book vector representations, further improving the quality of the conversion.

- Faster training: VQ layers are often faster than other image conversion methods. This is because the VQ layer only needs to learn a set of discrete codebook vectors, rather than an exact representation of each input vector, which significantly reduces training time.
- Interpretability: The VQ layer quantizes continuous data into discrete vectors by means of discrete codebook vectors, which makes the resulting sketches easier to interpret and understand.

Overall, the VQ layer offers excellent conversion quality, faster training and interpretability in the field of Face2Sketch, and is therefore of great value in practical applications.

IV. EXPERIMENTAL RESULTS

A. Experimental Setup

In this paper, we used 376 pieces of data from CUFS [9], 188 pieces of face data and 188 pieces of sketch data, and then cut the data into 150 pieces of face training data, 150 pieces of sketch training data, 38 pieces of face test data and 38 pieces of sketch test data in a ratio of eight to two. The training data was used to build a model and the test data was used to evaluate the model. Also, this paper opens up the data and deploys it to Git to facilitate direct connection and use of data through Google Colab.

CASQ is mainly built by Tensorflow [26] and Sonnet [27], which are well-known in the field of deep learning Python, with Tensorflow version 2.11.0 and Sonnet version 2.0.1.

The image size of the original data is 200*250*channels, the channel of the RGB image is 3, and the channel of the grayscale image is 1. It is found that there are often black traces on the corners of the data. It is speculated that it was left when the original data was created. Next, in order to ensure the input and output quality of the model, this paper cuts the size of the data to 200*200*channels, which not only retains the important part of the image, but also reduces the training burden of the model.

In the data preprocessing section, the vector of RGB images is between 0 and 255 and has three channels, while the vector of grayscale images is between 0 and 255 and has only one channel. We first add two additional channels to the sketch data, i.e., grayscale images, so that our input and output sizes are equal. Then we normalize the vector, that is, let the value of the vector be between 0 and 1. The reason for doing this is that if the input layer is large, the gradient passed to the input layer becomes large during back propagation. With a large gradient, the learning rate has to be very small, otherwise it will cross the optimum. In this case, the learning rate needs to be chosen with reference to the size of the input layer values, and a direct data normalization operation makes it easy to choose the learning rate.

In order to highlight the rapid adaptability and learning power of CASQ models, we standardize the parameters in this paper to facilitate comparing the differences between models at the same benchmark, and set the learning rate to 0.001, the batch size to 128, and the epochs to 1000.

1) DAE:

Figure 7 shows the model architecture of DAE in this paper. The optimizer is chosen as Adaptive Moment Estimation (Adam), which is a gradient descent based optimization algorithm that combines the momentum method and RMSProp method and is able to converge quickly and performs well in dealing with non-convex problems. The loss function part is chosen as a binary cross entropy, which measures the difference between the actual output value and the predicted output value, with smaller values indicating more accurate model predictions. In deep learning, we usually use back propagation algorithms to minimize the loss function and thus train the model to improve its accuracy.

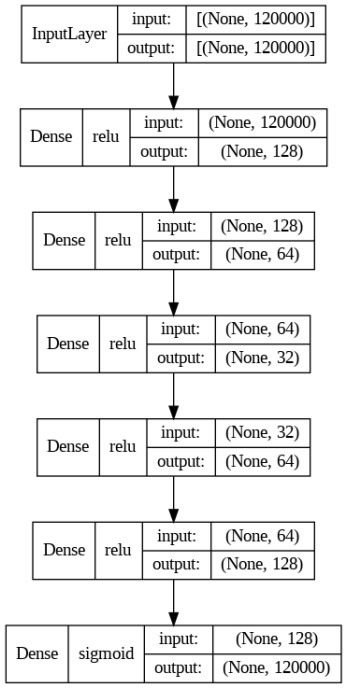

Fig. 7: DAE architecture in this paper

2) VAE:

Figure 8 shows the model architecture of VAE in this paper. The optimizer is also chosen as Adam. Unlike the DAE, a KL divergence loss is used to encourage the potential representation to obey the standard normal distribution, in addition to the binary cross entropy as the loss function. The total loss is a linear combination of the reconstruction loss and the KL divergence loss, and the ratio used in this paper is one to one, in other words, the total loss is the average of the reconstruction loss and the KL divergence loss.

3) CASQ:

Figure 9 shows the model architecture of CASQ in this paper. To facilitate model comparison, CASQ and DAE use the same optimizer and loss function, i.e., Adam and binary cross entropy

B. Training of My Method

1) DAE:

Based on Figure 10, it can be seen that there are several areas of loss amplification in the DAE during loss convergence.

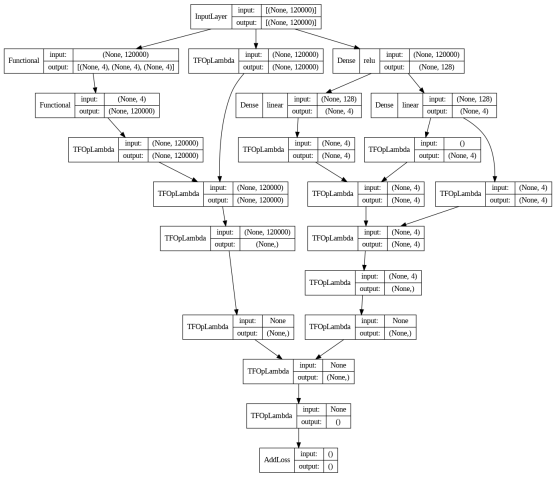

Fig. 8: VAE architecture in this paper

The possible reason for this is that the learning rate is set too large, causing the pace of each learning correction to move too far.

2) VAE:

In contrast to the DAE training, Figure 11 shows that the VAE loss is clearly converging and there is no problem with the excessive pace of correction. However, VAE is particularly important in terms of setting the starting position of the loss as the total loss calculation takes into account the KL divergence, and sometimes the model can be trained in such a way that the loss calculation is too large that it becomes impossible to calculate. In contrast, the VAE parameters are more reasonable, but more care needs to be taken in the first step of calculating the error

3) CASQ:

Because of the addition of the self-attention mechanism and quantization layer, CASQ needs to be more cautious in parameter setting. Figure 12 shows that the part of the loss amplification is a little more than that of DAE. As the loss of DAE and VAE decreases gradually, CASQ still has more room for improvement, and this is only 1000 epochs. It can be seen that CASQ has full potential in terms of learning adaptability and future performance.

C. Test Results for Comparison

SSIM (Structural Similarity Index) and MSE (Mean Squared Error) are both common image quality evaluation indicators.

SSIM is an image quality evaluation index based on structural similarity, which can be used to evaluate the similarity between two images. SSIM is based on the characteristics of human perception, it takes into account not only the brightness, contrast and structural information of images, but also the relationship between them, so it is more in line with the subjective feeling of human on image quality.

MSE is a pixel-based image quality evaluation metric that measures the square of the average pixel difference between two images; the smaller the MSE, the smaller the difference between the two images and the better the quality.

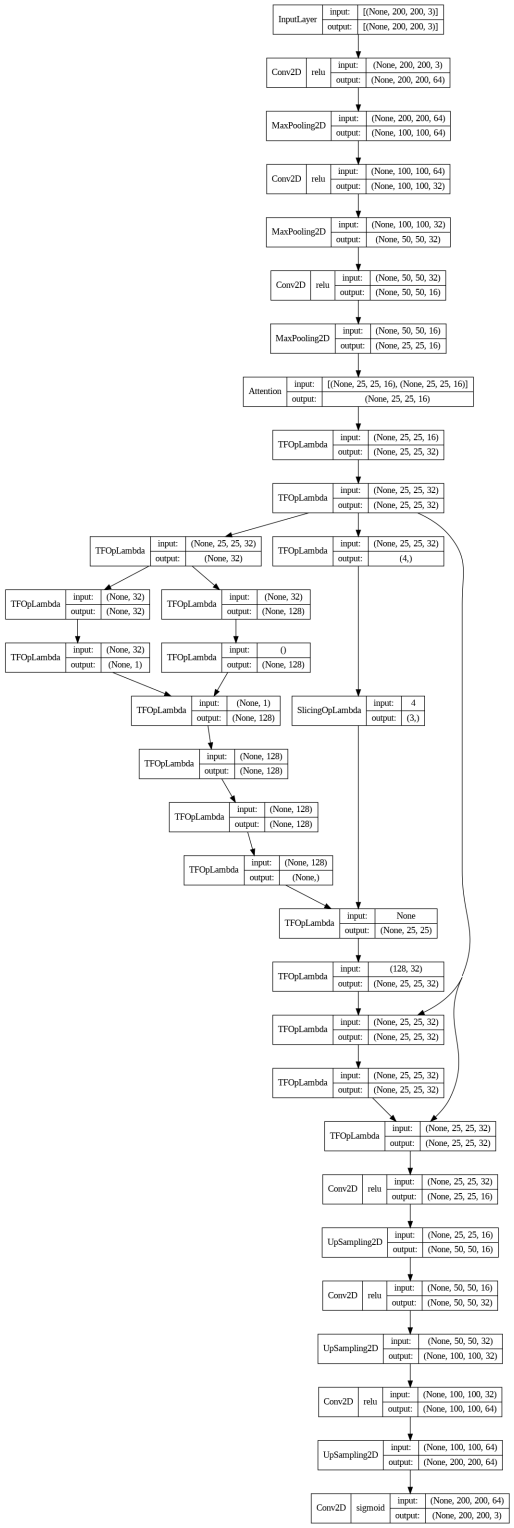

Fig. 9: CASQ architecture

Overall, both SSIM and MSE are used to evaluate image quality metrics, but they are based on different principles: SSIM is more in line with the subjective human perception of image quality, while MSE focuses more on the difference between pixel values [28].

As can be seen from Table I, CASQ obtained the best results

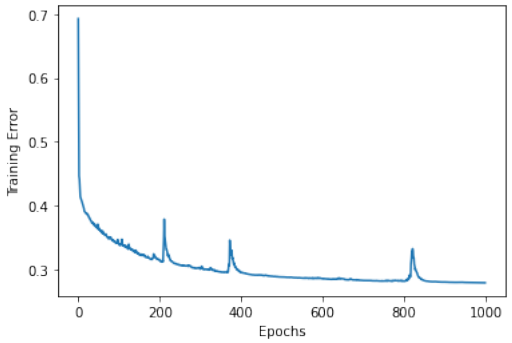

Fig. 10: Training loss diagram for DAE

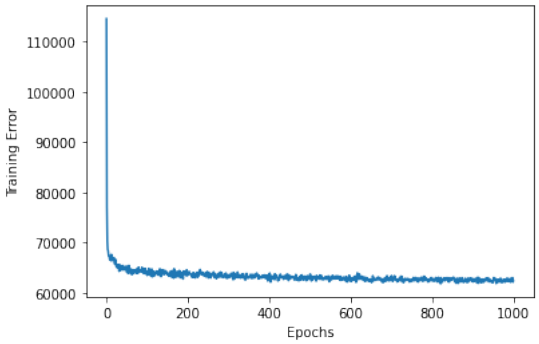

Fig. 11: Training loss diagram for VAE

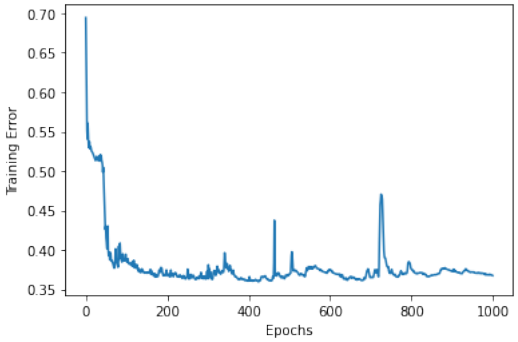

Fig. 12: Training loss diagram for CASQ

for any assessment index. That is to say, CASQ outperformed DAE and VAE, both in terms of subjective human perception of image quality and in terms of the difference between pixel values.

TABLE I: Image quality assessment

Models	MSE	SSIM
DAE	0.049908	0.471451
VAE	0.099079	0.470112
CASQ	0.031974	0.528875

V. DISCUSSION

In this paper, we compared the performance of three different models - DAE, VAE, and CASQ - for the task of Face2Sketch generation. Our results showed that CASQ outperformed the other two models in terms of both visual quality and quantitative evaluation metrics.

Our findings suggest that the use of self-attention and quantization techniques in CASQ is effective for capturing the complex features of faces and generating high-quality sketches. In particular, the self-attention mechanism allows the model to focus on important facial features and generate more detailed and accurate sketches. The quantization technique helps to reduce the memory footprint of the model and speed up the training process, while maintaining high performance.

However, there are some limitations to our study. First, we only evaluated the performance of the models on a single dataset, which may limit the generalizability of our results to other datasets. Second, we did not compare our results with state-of-the-art models for Face2Sketch generation. Future studies should evaluate the performance of CASQ on multiple datasets and compare its performance with other state-of-the-art models.

Despite these limitations, our study has important implications for the development of Face2Sketch generation models. CASQ has demonstrated superior performance compared to other models, and its use of self-attention and quantization techniques may be useful for other image generation tasks. Future research could explore the use of these techniques in other domains and evaluate their effectiveness.

VI. CONCLUSION

In conclusion, our study demonstrates the effectiveness of CASQ for Face2Sketch generation, and highlights the potential of self-attention and quantization techniques in image generation tasks. This work contributes to the field of computer vision by providing a new approach for generating high-quality sketches from facial images, which could be useful in a variety of applications, such as forensic art, digital entertainment, and facial recognition systems.

Future research could further optimize and refine the self-attention and quantization techniques used in CASQ, and explore their effectiveness in other image generation tasks. Additionally, the performance of CASQ could be compared with other state-of-the-art models on multiple datasets to further evaluate its effectiveness and generalizability.

Overall, our study provides valuable insights into the development of image generation models, and demonstrates the potential of self-attention and quantization techniques in this field. We hope that our findings will inspire further research in this area and contribute to the development of more advanced and effective image generation models.

The dataset used in this paper and the model written in Python are open-sourced to the following Github link: https://github.com/Dino1G/SCC413

REFERENCES

[1] W. W. Bledsoe and I. Browning, “Pattern recognition and reading by machine,” in *Papers presented at the December 1-3, 1959, eastern joint IRE-AIEE-ACM computer conference*, 1959, pp. 225–232.

[2] A. J. Goldstein, L. D. Harmon, and A. B. Lesk, "Identification of human faces," *Proceedings of the IEEE*, vol. 59, no. 5, pp. 748–760, 1971.

[3] L. Sirovich and M. Kirby, "Low-dimensional procedure for the characterization of human faces," *Josa a*, vol. 4, no. 3, pp. 519–524, 1987.

[4] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman, "Eigenfaces vs. fisherfaces: Recognition using class specific linear projection," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 19, no. 7, pp. 711–720, 1997.

[5] Y. Kortli, M. Jridi, A. Al Falou, and M. Atri, "Face recognition systems: A survey," *Sensors*, vol. 20, no. 2, p. 342, 2020.

[6] X. Lv, M. Su, and Z. Wang, "Application of face recognition method under deep learning algorithm in embedded systems," *Microprocessors and Microsystems*, p. 104034, 2021.

[7] W. Wan, Y. Gao, and H. J. Lee, "Transfer deep feature learning for face sketch recognition," *Neural Computing and Applications*, vol. 31, pp. 9175–9184, 2019.

[8] S. Yu, H. Han, S. Shan, A. Dantcheva, and X. Chen, "Improving face sketch recognition via adversarial sketch-photo transformation," in *2019 14th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2019)*. IEEE, 2019, pp. 1–8.

[9] X. Wang and X. Tang, "Face photo-sketch synthesis and recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 31, no. 11, pp. 1955–1967, 2008.

[10] H. Bi, N. Li, H. Guan, D. Lu, and L. Yang, "A multi-scale conditional generative adversarial network for face sketch synthesis," in *2019 IEEE international conference on image processing (ICIP)*. IEEE, 2019, pp. 3876–3880.

[11] D. Bank, N. Koenigstein, and R. Giryes, "Autoencoders," *arXiv preprint arXiv:2003.05991*, 2020.

[12] Q. Meng, D. Catchpoole, D. Skillicom, and P. J. Kennedy, "Relational autoencoder for feature extraction," in *2017 International joint conference on neural networks (IJCNN)*. IEEE, 2017, pp. 364–371.

[13] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[14] H. Tang, X. Chen, W. Wang, D. Xu, J. J. Corso, N. Sebe, and Y. Yan, "Attribute-guided sketch generation," in *2019 14th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2019)*. IEEE, 2019, pp. 1–7.

[15] X. Qi, M. Sun, W. Wang, X. Dong, Q. Li, and C. Shan, "Face sketch synthesis via semantic-driven generative adversarial network," in *2021 IEEE International Joint Conference on Biometrics (IJCB)*. IEEE, 2021, pp. 1–8.

[16] J. Hou, J. D. Deng, S. Cranefield, and X. Ding, "Cross-domain latent modulation for variational transfer learning," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2021, pp. 3149–3158.

[17] Z. Cheng, H. Sun, M. Takeuchi, and J. Katto, "Performance comparison of convolutional autoencoders, generative adversarial networks and super-resolution for image compression," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 2613–2616.

[18] J. Bao, L. Li, and A. Davis, "Variational autoencoder or generative adversarial networks? a comparison of two deep learning methods for flow and transport data assimilation," *Mathematical Geosciences*, vol. 54, no. 6, pp. 1017–1042, 2022.

[19] C. Zhou and R. C. Paffenroth, "Anomaly detection with robust deep autoencoders," in *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, 2017, pp. 665–674.

[20] K. G. Lore, A. Akintayo, and S. Sarkar, "Llnet: A deep autoencoder approach to natural low-light image enhancement," *Pattern Recognition*, vol. 61, pp. 650–662, 2017.

[21] X. Hou, L. Shen, K. Sun, and G. Qiu, "Deep feature consistent variational autoencoder," in *2017 IEEE winter conference on applications of computer vision (WACV)*. IEEE, 2017, pp. 1133–1141.

[22] Z. Cheng, H. Sun, M. Takeuchi, and J. Katto, "Deep convolutional autoencoder-based lossy image compression," in *2018 Picture Coding Symposium (PCS)*. IEEE, 2018, pp. 253–257.

[23] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[24] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, "Self-attention generative adversarial networks," in *International conference on machine learning*. PMLR, 2019, pp. 7354–7363.

[25] A. Van Den Oord, O. Vinyals *et al.*, "Neural discrete representation learning," *Advances in neural information processing systems*, vol. 30, 2017.

[26] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[27] Csuperbobry, "Sonnet," https://github.com/deepmind/sonnet, 2020.

[28] U. Sara, M. Akter, and M. S. Uddin, "Image quality assessment through fsim, ssim, mse and psnr—a comparative study," *Journal of Computer and Communications*, vol. 7, no. 3, pp. 8–18, 2019.