Investigative Study on Generative Models for Face2Sketch and Sketch2Face

Antoni Skubisz - 36145142

***Abstract*—Face sketch synthesis has gained a lot of popularity in recent years. To answer the demand for it, many deep learning models were studied to facilitate this process. In this report, we explore and compare four of them, including our novel transfer learning autoencoder which utilises VGG-16 network in the encoder part. Our models are all trained on the CUHK student face dataset which has been artificially increased by data augmentation. It was found that our transfer learning model gave competitive results for face to sketch task, and performed the best out of all considered models for sketch to face, giving 0.807 SSIM and 0.015 MSE.**

I. Introduction

There is a wide range of functionalities of face to sketch generation which varies from serious issues to more shallow ones. Firstly, it can be utilised by helping law enforcement when it is necessary to compare suspects to composite drawings made by an artist [1], as it is time-consuming to create a sketch for each suspect by hand. Having a well-trained model that generates a sketch from the mugshot can greatly increase the speed of this process. Another, more superficial, application is for digital entertainment. People like to apply different filters on their pictures and being able to achieve a sketch-like image of their face, without the need to pay for a professional artist, is definitely something that would entertain users of social media apps.

Moreover, the inverse approach (generating a coloured face from sketch) is also a great area of interest. The main utility of this model is the real face generation from a forensic sketch drawn by an artist, as it is often the case that the only form of evidence is an eyewitness statement. In such case, the investigators can have a better understanding of what suspects might look like [2]. Other applications might be in digital art used by artists to quickly get realistic faces from sketches of the faces.

Fig. 1: Example of faces with corresponding sketches

As the interest in face sketch synthesis (FSS) increased, many deep learning architectures have been studied to facilitate this process [3], [4]. The most common choice for the model class is Generative Adversarial Networks (GANs), due to their adversarial learning process. They consist of the generator and discriminator who simultaneously compete with each other. The goal of the former is to generate realistic images while not having access to the real images. It learns by interacting with a discriminator which has access to both real and fake images and its job is to learn to correctly identify the fake images [5]. The basic architecture of this model is illustrated in Figure 2.

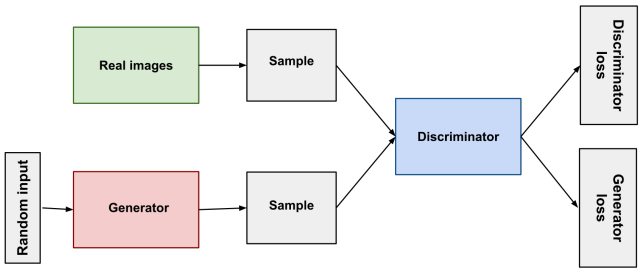

Fig. 2: GAN structure (image from [6])

Next, a less popular approach is using autoencoders. They are designed to encode the input into meaningful code (latent space) that can be decoded to resemble the input image. The main idea behind them is that encoder compresses the most important features of the input so that the decoder can learn to reconstruct the input. Their architecture usually follows the hourglass shape. Coupled with other neural network architectures such as convolutional neural networks or recurrent neural networks, the usefulness of autoencoders can be greatly extended to image generation or natural language processing [7]. Autoencoders have been widely used for problems such as denoising, anomaly detection, or dimensionality reduction [8].

Both of the above approaches can be further improved by using a concept called transfer learning. It relies on, as the name suggests, transferring the trained weights from the previous pre-trained model onto our new model as a 'starting point'. That pre-trained model usually is trained on a big dataset (ImageNet) so that it generally knows which features to extract. The weights can be later fine-tuned, to be more relevant to our task. This approach is particularly useful when our dataset is relatively small [9]. We can see an example of using transfer learning in the GANs architecture in the

Mind2Mind model [10].

In the next section we will review relevant literature and explain our motivation for choosing the specific methods. Next, in Section III we introduce and explain methodologies of sparse autoencoder, convolutional autoencoder, cGAN (pix2pix), and our new autoencoder which uses transfer learning. In Section IV we introduce our dataset and present the results of our models. Lastly, in the discussion and conclusion sections, we reflect on our findings and conclude the report.

II. Relevant work

A. Image to image

Face sketch synthesis task is regarded as an image-to-image translation problem. There are many methods related to this problem, with the most famous being pix2pix [11] and CycleGANs [12]. Pix2pix is a conditional GAN (cGAN) model that can be applied to a variety of tasks such as colouring black and white pictures or turning Google Maps photos into aerial images. Its architecture consists of a typical generator and discriminator, with the generator following U-Net structure [13] and the discriminator uses a convolutional PatchGAN classifier. CycleGANs introduce the usage of two generator networks and two discriminators. Each generator is responsible for learning a mapping from domain A to domain B and its inverse, meanwhile, each discriminator evaluates their accuracies. CycleGANs also introduce cycle consistency loss which makes sure that generated output can be transformed back into the input domain. In this report, we will use the pix2pix model as one of our benchmarks for comparison.

B. Face sketch synthesis

Since the increase in the utility of face-to-sketch translation, extensive efforts have been put into the research of the most effective methods. The most impressive results are yielded by some variations of GANs. For example, the use of composition-aided GANs (CA-GAN) [4] or back-projection GANs (BP-GAN) [14]. The former takes additional input of composition rules of different facial features which helps to generate images that are consistent with the set rules. The latter acts like a usual GAN with the difference being that the output gets projected into the input space using projection filters. This way, a generated sketch is similar to the input, and the model is able to produce more realistic and accurate output. However, there are also different approaches, for example, Bayesian framework [15]. The proposed method consists of neighbour selection and weight computation. This method was special because it includes spatial neighbouring with adjacent patches as a constraint. It means that the model is able to capture relationships between, for example, the position of the eyes and the eyebrows. Another approach that considers spatial dependencies is Markov random neural fields [16]. It relies on cooperation between Markov Random Fields (MRFs) and Neural Networks (NNs). MRFs are used to model those spatial features, meanwhile, NNs learn how to map the input to the output.

Most of the methods listed above achieve great results, similar to state-of-art models, with some image degradation. The motivation for this work is to explore and compare the usefulness of transfer learning in the autoencoder for the face sketch synthesis task. We hope that the pre-trained model will be able to pick up important facial features more easily and will be able to compete with the implementation of a well-known pix2pix model.

III. Methodology

As mentioned above, we will compare our transfer learning autoencoder, with some baseline models, which are going to be sparse autoencoder, convolutional autoencoder, and pix2pix. We will introduce and explain the theory behind each one of them.

A. Sparse autoencoder

Let's start by explaining what an autoencoder is. It is a type of neural network whose task is to learn to extract the most important features of the input and compress it into lower dimensional 'code', that can be later reconstructed back into its original form. The architecture of the autoencoders consists of two parts: encoder and decoder. Inside the encoder part, the input is getting scaled down with each consecutive layer. Those layers can be traditional fully connected ones, however, the performance can be improved by incorporating convolutional or LSTM layers as well. The main advantage of autoencoders is that they can be used for unsupervised learning as they do not need labeled data for training. Moreover, the input data requires no feature engineering which is useful when dealing with high dimensional data such as text, audio or images. Nevertheless, there are some things that need to be taken into consideration when using autoencoders. Firstly, they are prone to overfitting, especially when the structure of the network is too deep and complex, or when the dataset is too small. Secondly, autoencoders might be very computationally expensive to train as there is usually a lot of connection between neurons.

Now, let's introduce sparse autoencoders. Similarly to the usual autoencoders, they are also trying to minimise the difference between the input and the reconstructed output. However, they also try to ensure that the code is sparse, which means that only some of the neurons in the code are activated. This is achieved by including a regularisation term in the loss function. We visualise the structure of the implemented sparse autoencoder in Figure 3.
From the graph we see that the input image is first flattened to a one-dimensional vector. Then, in the encoder part, it gradually gets connected with fewer amount of neurons, until it reaches the code part, where L1 regularisation is applied. Then the decoder part is symmetric to the encoder. Lastly, we use reshape layer to transform it back into an image. To summarise, the sequence of the number of neurons in each layer is equal to: 512, 256, 128, 64, 128, 256, 512. The model contains roughly 201 million parameters so it is quite computationally heavy. The main advantage of using a sparse

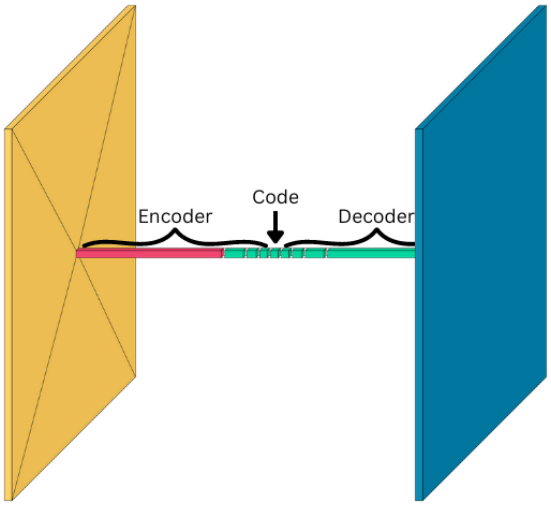

Fig. 3: Sparse autoencoder architecture

autoencoder is that, by encouraging sparsity in latent space, irrelevant or noisy features are filtered out which improves generalisation of the model.

B. Convolutional autoencoder

The next model that we will compare is the convolutional autoencoder. We expect it to perform better than the sparse autoencoder as the usage of convolutional layers makes it more suitable for the image type data. The advantage of using them over traditional dense layers is that they are able to learn local and global patterns in the image, even including translated variations of it. Moreover, they are able to do so with much fewer parameters. We present the general structure of our implemented convolutional autoencoder in Table I.

Layer	Channels	Size	Kernel size	Stride	Padding
Input	3	256x256	-	-	-
Conv2D	16	128x128	5x5	2	same
Conv2D	32	64x64	5x5	2	same
Conv2D	64	32x32	5x5	2	same
Conv2DTranspose	64	64x64	5x5	2	same
Conv2DTranspose	32	128x128	5x5	2	same
Conv2DTranspose	16	256x256	5x5	2	same
Conv2D	3	256x256	5x5	1	same

TABLE I: Convolutional autoencoder architecture

It can be noted that we did not use any pooling layers in the architecture. It is due to the fact that using them led to blurry output during experiments. Instead, we decided to downsample the image using a stride equal to 2, which made input halve after each encoding layer, and double in the decoding part. Such choice of number of filters in convolutional and deconvolutional layers was found to be optimal. Even though it was feasible to include more layers with more filters, the results were much worse, probably due to overfitting. In the table, we also see that filter size, size, and padding were all kept constant in the entire architecture. What has not been included in the table is the activation function which we chose to be LeakyReLU with an alpha parameter equal to 0.1. The main advantage of LeakyReLU over standard ReLU is that due to sparsity induced by ReLU, some neurons become "dead" and do not contribute to the model. Leaky ReLU solves this problem by including some really small slope in the negative region, which ensures that all neurons are active. We visualise the model in Figure 4. It can be seen there that our autoencoder is not perfectly symmetric.

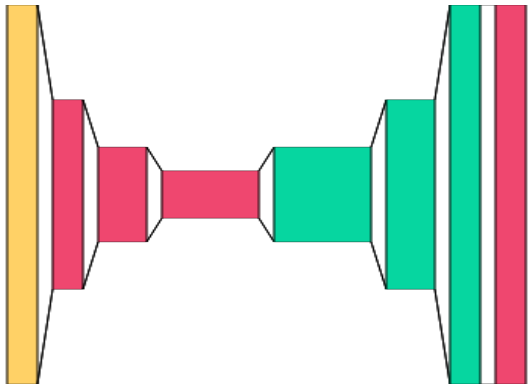

Fig. 4: Convolutional autoencoder architecture

C. Pix2pix

Now, to compare our models with some state-of-art architecture we found pix2pix keras implementation on the internet (https://machinelearningmastery.com/how-to-implement-pix2pix-gan-models-from-scratch-with-keras/). Pix2pix is a type of conditional GAN, so it consists of generator and discriminator. The generator is built according to U-Net architecture. Each encoding block follows the structure:

$$\text{Conv2D} \rightarrow \text{Batch Normalisation} \rightarrow \text{LeakyReLU}$$

meanwhile, the decoding block:

$$\text{Conv2DTranspose} \rightarrow \text{Batch Normalisation} \rightarrow \text{Dropout}^{*} \rightarrow \text{ReLU}$$

* Dropout is only applied in the first 3 decoding blocks.

The problem that was pointed out in the pix2pix paper [11] is that in the usual encoder-decoder networks the information needs to pass through all the layers including latent space. This can lead to the loss of some important features during the encoding process, resulting in poor quality output. The proposed solution is to use skip connections. They work by connecting a layer in the encoder with the corresponding layer in the decoder. Such connection allows the transfer of learned features across the autoencoder, which helps to keep the details in the output. In pix2pix the skip connections were applied between each layer i and $n - i$, where n is the total number of layers.

Discriminator in pix2pix follows architecture introduced in the same paper [11], called PatchGAN. It was motivated by the observation that L1 and L2 fail to encourage high level details. Because of that, the proposed solution was focused to ensure high level correctness while leaving the low level features to rely on the L1 term. The main idea behind the

PatchGAN is to, instead of classifying the entire picture, look at each $N \times N$ patch of the image, classify them separately, and average the result to get the final decision if the image is fake or real. Such design ensures that attention to high-level details is kept, rather than the overall structure.

The general advantages of pix2pix are its flexibility and the quality of the output. If provided with the dataset, pix2pix can be applied to a wide range of applications, while providing high quality output. Nevertheless, there are some shortcomings. Essentially, pix2pix model is very expensive to train. One epoch of training on our dataset took around 1 hour 30 minutes. Moreover, outputs from that model are usually very similar to the input image, hence it is slightly limited.

D. Transfer learning autoencoder

Now, we introduce our autoencoder that uses transfer learning in the encoder part. We decided to use VGG-16 convolutional neural network as our pre-trained model. The rationale behind it was that it is a very deep model that has been trained on over 14 million images. Hence, it should have learned a wide range of features that can be useful for our task, mainly those face related. It is also very easy to implement due to existing libraries inside of keras. The main idea behind VGG-16 is to stack many convolutional layers with small 3×3 filters on top of each other. Convolutional layers are used to increase the number of channels, meanwhile, max pooling layers decrease the size of the image. Padding, stride, and filter size are kept constant throughout the VGG-16 network. Fully connected layers have not been included in our autoencoder.

In the decoder part, we only used transposed conv2D layers due to our experience with convolutional autoencoder where upsampling layers gave worse results. The number of channels in the decoding layers was chosen according to the inverse order present in the VGG-16. Detailed architecture is shown in Table 5.

Layer	Channels	Size	Kernel size	Stride	Padding
Input	3	224x224	-	-	-
2xConvolution	64	224x224	3x3	1	same
Max-Pooling	64	112x112	3x3	2	valid
2xConvolution	128	112x112	3x3	1	same
Max-Pooling	128	56x56	3x3	2	valid
3xConvolution	256	56x56	3x3	1	same
Max-Pooling	256	28x28	3x3	2	valid
3xConvolution	512	28x28	3x3	1	same
Max-Pooling	512	14x14	3x3	2	valid
3xConvolution	512	14x14	3x3	1	same
Max-Pooling	512	7x7	3x3	2	valid
Conv2DTranspose	256	14x14	5x5	2	same
Conv2DTranspose	128	28x28	5x5	2	same
Conv2DTranspose	64	56x56	5x5	2	same
Conv2DTranspose	32	112x112	5x5	2	same
Conv2DTranspose	16	224x224	5x5	2	same
Conv2DTranspose	3	224x224	5x5	1	same

TABLE II: Transfer learning autoencoder architecture

Activation function used in encoder part is ReLU, as in the original VGG-16, however, in the decoder, we decided to use an alternative LeakyReLU for the reason similar as in convolutional autoencoder. A combination of Adam optimiser with MSE loss function was found to yield the best results. The other possible choices that were checked are SGD and RMSprop optimisers and binary cross-entropy loss function. The final model contained close to 16.3 million parameters, where 14.7 million were frozen weights from a pre-trained VGG-16 model. The usual practice when training a transfer learning model is to perform fine-tuning. This is the last step, where we unfreeze all the weights and re-train the whole model for a few epochs. This is done to adapt the pre-trained features for our new dataset. However this can lead to very quick overfitting, hence it is usually done with a very low learning rate. In our case, we performed it for 10 epochs with a learning rate of 0.0001. The visualisation of our autoencoder is shown in Figure 5.

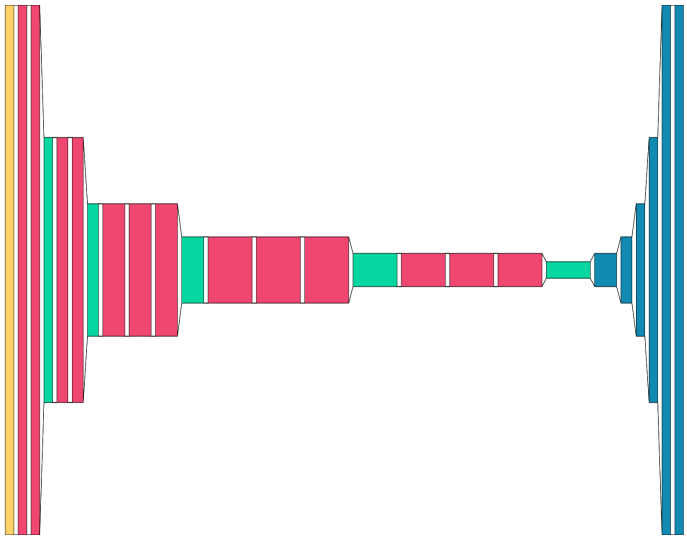

Fig. 5: Transfer learning autoencoder architecture

IV. Experiments

A. Experimental setup

The dataset that has been used for training of our models is the CUHK Student Face dataset available from the link provided in the coursework brief. This data contains 188 photos of faces of Asian students, paired with professional sketches. Since it is a relatively small sample, we decided to perform data augmentation. In essence, we artificially increased the size of our dataset by rotating and flipping horizontally and vertically. We took this idea from Kaggle notebook (https://www.kaggle.com/code/theblackmamba31/photo-to-sketch-using-autoencoder). Consequently, we ended up with 1504 photos and corresponding sketches. Each picture is resized to 256×256 pixels (with an exception for transfer learning autoencoder where the size is set to 224×224). To make our experiment more reliable we randomly split our data into test and training sets in roughly 80:20 proportion (300 in the test set, 1204 in the training set). Each autoencoder model is trained for 100 epochs with a batch size of 32 and the default loss function being MSE. However, when running experiments for sparse autoencoder, binary cross-entropy was found to yield better results, hence it was our choice in that specific case. Each pix2pix model (face to sketch and

sketch to face) got trained on only 6 epochs, with a batch size of 1, (as in the pix2pix paper) due to the time and computational constraints. The whole experiment was run on GPU accelerated google collab environment.

When comparing our predictions to the test set, other than eye evaluation, we also need some formal mathematic measures to calculate similarity. The ones that we chose due to the simplicity of their implementation are Structural Similarity Index Measure (SSIM) and Mean Squared Error (MSE). The former ranges from -1 to 1 with 1 meaning that pictures are identical. The formula for it takes into consideration three aspects: luminance, contrast, and structure. MSE is an average squared distance between each predicted pixel and the corresponding ground truth pixel. The drawback of this measure is that large errors are penalised more heavily.

B. Face to sketch results

First, we present the results of our models for face to sketch task. To train them, we use photos from the training set as input, and sketches as the target variable. Performance on test data is shown in Figure 6, with the first column being the original sketch, followed by sparse autoencoder, convolutional autoencoder, pix2pix, and transfer learning autoencoder.

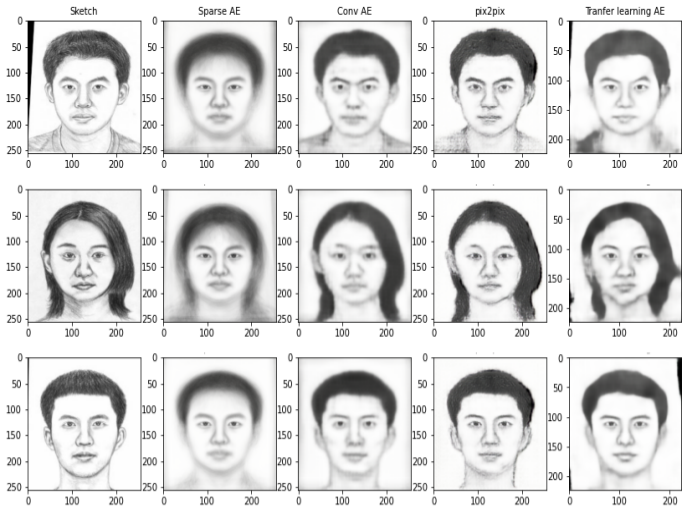

Fig. 6: Predictions of our models on F2SK task

As expected, the sparse autoencoder did the worst out of the four models. Only the main components such as head and hair were reconstructed but with no details. Moreover, the features themselves are very blurry. The rest of the models did reasonably well, depicting face details accurately. Our transfer learning model did sometimes exhibit some face degradation which can be seen in the second row. Hair texture seemed to be best captured by the pix2pix model. More formal similarity measures have been calculated over 300 test pictures and aggregated results are shown in Table III.

Error metric	Sparse AE	Conv AE	pix2pix	Transfer learning AE
SSIM	0.699	**0.768**	0.744	0.746
MSE	0.034	**0.022**	0.029	0.028

TABLE III: Error metrics on F2SK task

Surprisingly, convolutional autoencoder has shown the best results for both similarity measures. Human eye judgment would possibly say that pix2pix gives more detailed and sharp sketches. It is worth noting that our transfer learning model, even though showed some image degradation, has achieved a slightly better result than the pix2pix model. However, measures such as SSIM and MSE have to be interpreted with caution.

C. Sketch to face results

Next, we switch the task to predicting a coloured photo from a sketch. We show results in a similar fashion as for F2SK, presenting the same people from the test set (Figure 7).

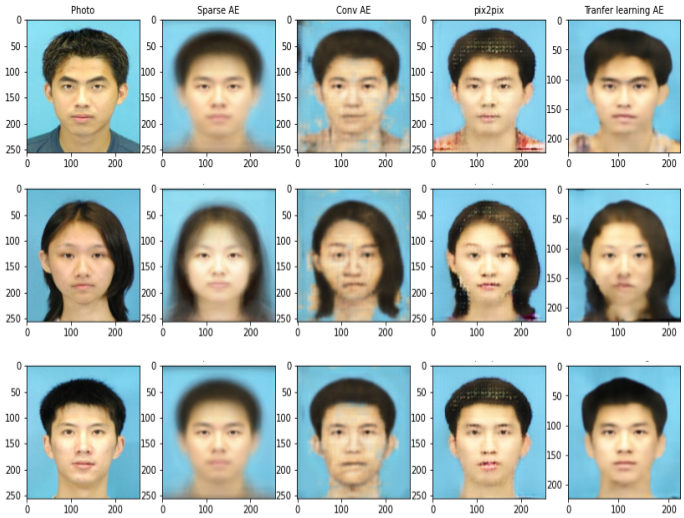

Fig. 7: Predictions of our models on SK2F task

Similarly, as in face to sketch, sparse autoencoder outputs very generic faces with first and third row images being fundamentally the same. Convolutional autoencoder, even though was very accurate for F2SK, here showcases very visible degradation with many pixels around the face being wrongly interpreted. Pix2pix gave satisfactory results, however, the outputs have some unexpected white pixels that are clearly visible in the hair. Our transfer learning autoencoder, on the other hand, did not have any of those problems and the output looks smooth. The only issue is that in the second row, the face is slightly deformed. All the models showed some problems with accurately capturing the colour of the shirt or hair colour, which is expected as it is hard to deduce colours from the sketch. As in F2SK we calculate similarity measures over 300 test pictures and show the average values in Table IV.

Error metric	Sparse AE	Conv AE	pix2pix	Transfer learning AE
SSIM	0.76	0.782	0.774	**0.807**
MSE	0.028	0.02	0.021	**0.015**

TABLE IV: Error metrics on SK2F task

Results in the table confirm our observations, the transfer learning autoencoder did the best with significant difference over second-best model.

V. Discussions

During the experiments, we noticed that our novel transfer learning autoencoder did overall the best job across other considered models. Although the convolutional autoencoder did better for the F2SK task, the difference in similarity measures was insignificant. On the other hand in SK2F, our model did substantially better. It is worth noting that similarity measures considered in this report might be misleading and not entirely accurate as they are sensitive to slight pixel-level errors. Meanwhile, the human eye does not detect those granular mistakes and focuses more on the general structure and details [17]. It would be interesting to use more complicated similarity measures such as FID, FSIM, or SCOOT, however due to the time constraints and lack of easily accessible Python library implementations we decided to focus on SSIM and MSE. Future improvements might include using a transfer learning model that has been pre-trained on face datasets rather than a very general one like ImageNet.

VI. Conclusion

To summarise, we attempted the task of face sketch synthesis. Four different models have been explored including three autoencoders (one novel involving transfer learning) and one conditional GAN. Inside autoencoders, we introduced our own number of layers and neurons. The code with all the trained models and dataset is available on our Google Drive folder. The similarity measures have been found for each model and we concluded that our novel transfer learning model did the best job overall for both face to sketch, and sketch to face tasks.

References

[1] N. Balayesu and H. K. Kalluri, "An extensive survey on traditional and deep learning-based face sketch synthesis models," *International Journal of Information Technology*, vol. 12, no. 3, pp. 995–1004, 2020.

[2] S. Klum, H. Han, A. K. Jain, and B. Klare, "Sketch based face recognition: Forensic vs. composite sketches," in *2013 international conference on biometrics (ICB)*. IEEE, 2013, pp. 1–8.

[3] L. Zhang, L. Lin, X. Wu, S. Ding, and L. Zhang, "End-to-end photo-sketch generation via fully convolutional representation learning," in *Proceedings of the 5th ACM on International Conference on Multimedia Retrieval*, 2015, pp. 627–634.

[4] J. Yu, X. Xu, F. Gao, S. Shi, M. Wang, D. Tao, and Q. Huang, "Toward realistic face photo–sketch synthesis via composition-aided gans," *IEEE transactions on cybernetics*, vol. 51, no. 9, pp. 4350–4362, 2020.

[5] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath, "Generative adversarial networks: An overview," *IEEE signal processing magazine*, vol. 35, no. 1, pp. 53–65, 2018.

[6] [Online]. Available: https://developers.google.com/machine-learning/gan/gan_structure

[7] V. Shankar and S. Parsana, "An overview and empirical comparison of natural language processing (nlp) models and an introduction to and empirical application of autoencoder models in marketing," *Journal of the Academy of Marketing Science*, vol. 50, no. 6, pp. 1324–1350, 2022.

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

[9] K. Weiss, T. M. Khoshgoftaar, and D. Wang, "A survey of transfer learning," *Journal of Big data*, vol. 3, no. 1, pp. 1–40, 2016.

[10] Y. Frégier and J.-B. Gouray, "Mind2mind: transfer learning for gans," in *Geometric Science of Information: 5th International Conference, GSI 2021, Paris, France, July 21–23, 2021, Proceedings 5*. Springer, 2021, pp. 851–859.

[11] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1125–1134.

[12] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2223–2232.

[13] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*. Springer, 2015, pp. 234–241.

[14] N. Wang, W. Zha, J. Li, and X. Gao, "Back projection: An effective postprocessing method for gan-based face sketch synthesis," *Pattern Recognition Letters*, vol. 107, pp. 59–65, 2018.

[15] N. Wang, X. Gao, L. Sun, and J. Li, "Bayesian face sketch synthesis," *IEEE transactions on image processing*, vol. 26, no. 3, pp. 1264–1274, 2017.

[16] M. Zhang, N. Wang, X. Gao, and Y. Li, "Markov random neural fields for face sketch synthesis." in *IJCAI*, 2018, pp. 1142–1148.

[17] D.-P. Fan, S. Zhang, Y.-H. Wu, M.-M. Cheng, B. Ren, R. Ji, and P. L. Rosin, "Face sketch synthesis style similarity: a new structure co-occurrence texture measure," *arXiv preprint arXiv:1804.02975*, 2018.