Using Autoencoders for face2sketch and sketch2face

Applied Data Mining Coursework 1

36071280
School of Computing and Communications
MSc. Data Science, Data Mining
Lancaster University
p.thirthahallivenkatesh@lancaster.ac.uk

***Abstract*—Given you have the face photos and their corresponded sketch images, you are required to design and train at least 3 different deep autoencoders or GANs, and test each for two generative tasks 1) Generate facial sketch from photo 2) Generate facial photos from sketch. Your report needs to explain three methods/models in theoretical analysis and claim any novel design you developed by yourself. By tests, you need to explain which model achieve the best and why.**

***Index Terms*—Auto-encoder, GAN, layers**

I. Introduction

In the world of social media and digital entertainment, Face photo-sketch synthesis has seen a steady growth in popularity. Many individuals enjoy using sketches as profile photos or avatars to represent their digital identity. It is a vital tool in the art and design community helping artists, animators etc. draw inspiration from real people to generate their animated versions making photo-sketch synthesis (face2sketch) and matching are important and practical problems in entertainment and multimedia industry. [8]

The digital entertainment space has some amusing but indispensable applications for sketch generation - face sketch morphing among images captured in different lighting conditions, multiview face sketch synthesis (FSS) and manipulation of facial expression in photos [21]. Sketch generation are also useful for developing facial caricature generator system such as PICASSO as designed by Brennan [3].

One of the important uses of sketches comes in surveillance law enforcement. Automatic retrieval of photos of the suspects from the police mug shot database can help narrow down potential suspects quickly. However, in most cases, the photo image of a suspect is not available. The best substitute is a sketch drawing based on the recollection of an eyewitness. [18] However, psychological studies on show that humans process the face as a complete, holistic entity rather than a composition of features. It is difficult for victims to accurately describe individual features, making sketch based face recognition hard. [4]. An accurate sketch2face model can assist the victims in correcting and improving the suspect's image before running it through known criminal databases to identify them sooner. It can also help in better recognition and lower false accusation rates when released to the public.

With current technology, the most advanced deep learning techniques that are highly competitive and competent for data generation are Generative Adversarial Networks (GAN) and Variational Autoencoder (VAE) [14].

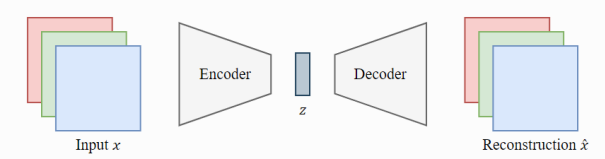

Fig. 1. Autoencoder

GANs are networks that contain two models - generative model and classifier model - and learn by pitting the former against the latter. The generative model uses noise as input and is trained on training data (images in this case) and generates new data. The classifier model tries to distinguish between the real and generated data. The entire model is trained until the classifier is mostly unable to distinguish between the real and generated data.

Autoencoders are fundamentally neural networks that are trained to attempt to copy input to output. Goodfellow et. all maintain that "Autoencoders can be thought of as being a special case of feedforward networks, and may be trained with all of the same techniques" [11]. Fig. 1 shows the basic design of an autoencoder. They can extract important features from the data and are an extension of dimensionality reduction techniques. Autoencoders can be built to extract the from an image and can be used for both sketch2face and face2sketch tasks.

This work explores different autoencoder models for generation of sketch images from actual photos and prediction (or generation) of facial images with color from black and white sketch images using models trained on these photos.

Section II details the extensive literature review undertaken for the research work in this project. Section III explains the methods used in this paper. Different types of autoencoders (namely Deep, Convolutional and Variational) were experimented with for this paper and the theories behind these are explained in Section III-A-III-C. The results from these experiments are shown in Section IV. The similarity indexes used to determine the accuracy of the model are detailed in Section IV-D.

II. Literature Review

The current main challenge in facial and sketch synthesis comes from the gap between the sketch image and actual photo.

The use of edge details extracted from an image using edge detection techniques and transform a particular image into a sketch is explored in Chandesa et. al [5]. They explore multiple filters for edge detection techniques and two image segmentation techniques - line thinning algorithm and 'bugwalk' for edge tracing. Laplacian of Gaussian (LoG) was polled as best edge detection technique in a survey of 80 human respondents to transform an image to sketch.

Gao et. al [22] propose an automatic FSS algorithm with local strategy based on embedded hidden Markov model (E-HMM) and selective ensemble which gives a decent result.It has moderate complexity and has good ability to extract two-dimensional facial features. The final image synthesized by the ensemble system is decent but has several flaws like bluriness, block edges etc. due to patching up of features instead of a complete image generation.

Zhang et. al. [1] note that sketch images can have exaggerated features as compared to real images and can distort the feature distributions on the facial points in the CUHK dataset [19]. This difference is identified as *synthetic gap* and the paper proposes a new framework termed Stacked Multichannel Autoencoder (SMCAE) which can fill the synthetic gap to better simulate real data. [1]

Another proposed deep learning framework consists for Sketch-to-Face Image generation consists of three modules in the system - Component Embedding (CE module) that uses an auto-encoder to learn key components of facial image, Feature Mapping (FM) and Image Synthesis (IS) modules that together form sub-network for mapping features and generating new images [6].

Xing et. al [7] propose a combination of deep Conditional Variational Autoencoder (CVAE) and Generative Adversarial Networks (GANs) called Attribute2Sketch2Face framework, which consists of three stages: (1) Synthesis of facial sketch from attributes using a CVAE architecture, (2) Enhancement of coarse sketches to produce sharper sketches using a GANbased framework [based on novel AUDeNet - UNet architecture with Dense network], and (3) Synthesis of face from sketch using another GAN-based network.

SegNet (Segmentation Network) proposed by Badrinarayan et. al. [2] show a Deep Convolutional network with a novel decoder model that uses pooling indices calculated in the encoder model to upsample non-linearly. This reduces training time greatly and achives good image segmentation.

The most advanced study on Super-resolution images performed by Saharia et.al. [17] and published in 2021, was highly successful in creating high-resolution images via stochastic denoising process. The model, termed SR3, was able to convert low resolution (64x64 images) to high-quality 1024x1024 images and showed high 'fool rate' (user studies that indicate the photo-realistic abilities of the synthetic images) of around 43%. This surpasses the previous record holding algorithm PULSE, proposed in 2020 by [15]. The training process of PULSE included traversing the high-resolution image and finding images that downscale. These high-resolution images have many applications in photo engineering, video streaming, video conferences etc. But these are still far away from being perfect models. As the resolution of the output image is increased, it is easier to spot blurs, improper edges, incorrect color mixing etc. for human to see and the fool rate of higher-resolution is still low.

The motivation of this study is to primarily understand the inner workings on some very common autoencoders and see their performance on images. A comparative study of encoder outputs with varying hyperparameters to see the change in performances is the goal of this experiment. Deep, Convolutional and Variational Autoencoders are selected because of the different techniques of learning image features and making predictions.

III. METHODOLOGIES

A. Deep Autoencoder

The first autoencoder in this experiment is a Dense Deep autoencoder which is basically feed-forward network. It uses multiple Dense layers in the encoder model to downsample the input image to the latent space of a given dimension and a same number of dense layers in decoder model to upsample to the output image.

A Dense layer in a neural network simply implies that all the neurons in this layer receive input from all the neurons in the previous layer.

Deep autoencoder processes the images as a flattened array of pixels rather than a multi-dimensional image. This method of processing images fails to consider the similarities in neighbouring pixels over multiples rows as each pixel is taken as input neuron.

B. Convolutional Autoencoder

Convolutional Autoencoder extends the previous autoencoder by using convolution layers instead of fully connected layers.

Convolution is simply an linear multiplication operation that works works by applying a filter (also referred to as kernel), which is basically a set of weights, over a section of an image to get a weighted average of that portion and resulting in activation of some important features of the image. Repeated convolutions over the entire image leads to generation of feature maps that indicate the location and intensity of the features.

Pooling layer essentially does a summarization operation to further reduce the dimensionality of the data. The most common statistical operation for pooling is max and thus, MaxPooling layer is commonly used in convolutional models. It has parameter such as stride - to control the pixel shift over the image, padding - to control the addition of extra pixels at the corners of the image to preserve the size of image from

one layer above, pool size - to define the size of the kernel etc.

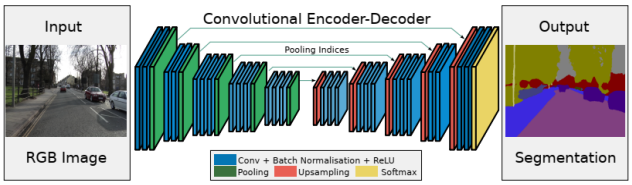

Fig. 2. Convolution Autoencoder- SegNet architecture

Fig. 2 show the architechture of SegNet - a convolutional autoencoder proposed by [2]. As can be seen, a decoder upsamples the input using transferred pool indices to produce a sparse feature map(s). Then convolution is performed with a trainable filter bank to make the feature map more dense. The final decoder output feature maps are fed to a soft-max classifier for pixel-wise classification [2].

C. Variational Autoencoder

An issue with standard autoencoders is that they encode each input datapoint independently. But if the input datapoints are of the similar type, there is a high chance of a relationship between the encoded versions of datapoints. This underlying similarity is usually not considered in most autoencoders since they do not follow a pre-defined distribution to encode the data [16]. Variational Autoencoders are a form of generative modelling type of machine learning model that focus on the distributions of the data, defined over the datapoints, in an N-dimensional space.

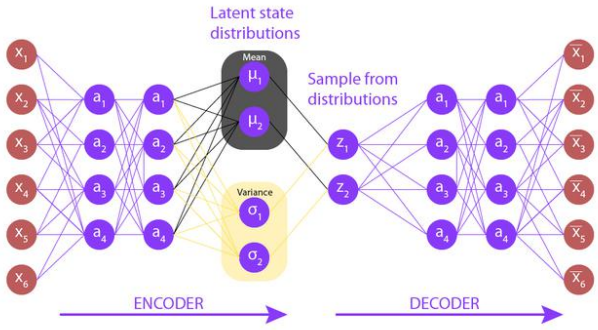

Fig. 3. Variational Autoencoder

The training dataset with X samples and P(X) distribution is used as the base to sample from and generate new data with similar P(X) distribution. The network outputs are parameters describing a distribution for each dimension in the latent space [12]. Figure 3 shows the design of this model. The distribution of the input data is assumed to be Gaussian and two vectors are given as encoder output that describe the mean and variance of the latent state distributions. The decoder model uses these distributions to sample and generate new images. Because a normal distribution is characterized based on the mean and the variance [16], VAE calculates both for each sample and checks that it follows a standard normal distribution (centered around 0).

Due to the relatively small size of facial photo-sketch datasets, the deep convolution network cannot extract enough effective and similar features from photos and sketches for recognition, a deep learning algorithm is needed to calculate probability of data distribution from the data [9]. This is an important consideration for choosing the appropriate models for face2sketch and sketch2face.

IV. Experimental Results

A. Experimental Setup

Details about setup, You may include a discussion on similarity measures briefly, with details how these measures are defined.

The experiment was run on CUHK Face Sketch database (CUFS) collected by Wang et. al [19]. This paper uses the 188 images from CUHK database. Several operation were performed on the imported images before passing to the autoencoder models. Only 80% of the available dataset is used for training purposes and the rest is used for testing the autoencoder models. Both sketch2face and face2sketch flows are trained on different instances of the same model to get the best results.

The images were read using OpenCV which stores the data in BGR format. To make it work with other libraries which hold the data in RGB format, we first transformed the image to the RGB colorspace using keras *cvtColor*. The images are resized to 256 x 256 size and normalize the values of the three channels.

Parameter	Value
aspect ratio	1
image size	256
image dimension	3
dropout	no
learning rate	0.001
epochs	500
training size	131
testing size	57

TABLE I
HYPERPARAMATER LIST

Loss functions are used to evaluate the discrepancy between the prediction and the real value, in order to optimize the models. A decrease in the loss calculated as per the loss function indicates an improvement in the performance of the model. Loss functions used in this experiment are - Mean absolute error and Binary cross entropy.

B. Discussion on Similarity Measures

Objective methods for assessing perceptual image quality traditionally attempted to quantify the visibility of errors (differences) between a distorted image and a reference image using a variety of known properties of the human visual system that is highly adapted for extracting structural information from a scene [20]

Some of the simplest metrics are -

1) Mean Absolute Error (MAE): MAE is a relatively simple metric that calculates the absolute value of the difference between the pixels in the two images show by Eqn. 1

$$MAE(x,y) = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n} \quad (1)$$

where y_i is the predicted value and x_i is the original value.

2) Peak signal-to-noise ratio (PSNR): It measures the ratio between maximum power of signal values and the maximun noise values in the image. It is an extension to the Mean-Squared error calculation which is computed by average of squared intensity differences of both image pixels given by Eqn. 2.

$$MSE(x,y) = \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} (x_{ij} - y_{ij})^2 \quad (2)$$

The PSNR takes a log of MSE and maximum pixel value of the image.

$$PSNR(x,y) = \frac{10 \log_{10}[\max(\max(X), \max(y))]^2}{|x-y|^2} \quad (3)$$

A higher value of PSNR indicates a higher image strength i.e, better image quality, with low noise inputs.

But these metrics do not consider the underlying structure of the images while measuring similarity. Wang et. al. [20] proposed the a new metric to measure the similarity between two given images.

3) Structured SIMilarity Index (SSIM): The index uses three comparisons between two images - luminance, contrast and structure. It works locally on image regions based on the window of Gaussian weight function used to calculate the values for the three components as shown in Eqn. (4)

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \quad (4)$$

where μ_x, μ_y are the mean intensities are the estimates of signal luminance, σ_x, σ_y are the standard deviations that represent the estimates of the signal contrast and C_1, C_2 are constants.

To get the SSIM index value over the entire image, on needs to take the mean of all the local SSIM values.

C. Environmental Setup

All models were trained on Google Colab. The final results presented in this paper were obtained by training without accelerated GPU support. Table III lists out the specifications of the training environment.

D. Results for Comparison

In present library implementations, optimizers can broadly classified into two types - gradient descent optimizers and adaptive optimizers [10], based on the necessary learning-rate hyperparameter. Adaptive algorithms automatically adapt the learning rate whereas in gradient descent algorithms it is fixed and user-defined. This experiment used one of each kind - Stochastic Gradient Descent and RMSProp (Root Mean Squared Propagation). Adam optimizer is also chosen for this experiment as it is regarded as the best performing optimizer for image datasets [13]. All the optimzer are implemented from the keras library.

Parameter	Value
Processor Count	2
Processor vendor	GenuineIntel
Processor Model name	Intel(R) Xeon(R) CPU @ 2.20GHz
CPU MHz	2200.216
CPU cores	1
Cache size	56320 KB
Total Memory available	13302920 kB

TABLE II
ENVIRONMENT SPECIFICATIONS

Note - This images and graphs recorded in this paper are imported from latest execution of the code and can have minor differences due to different training set and test set based on randomness during split.

1) Deep Autoencoder: The deep autoencoder in this experiment consists of 4 layers in the encoder with decreasing neuron sizes from SIZE in the first layer, SIZE/2 in the 2nd layer, SIZE/4 in the 3rd layer and finally compressed to SIZE/8 in the last layer. The information is stores in a latent space of 32 dimensions. The decoder network follows the same sizes in other direction.

After training with different optimizers and loss functions, the deep auto encoder performed best with Adam optimizer and Mean Absolute Error loss function. This model was trained for 200 epochs. The images generated by the sketch2face Deep model are shown in

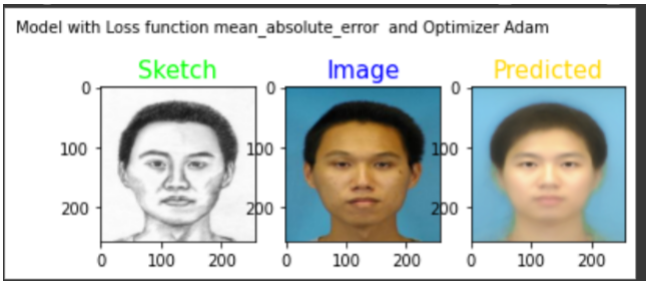

Fig. 4. Predictions of Deep Autoencoder sketch2face

The images generated by the face2sketch Deep model are shown in .

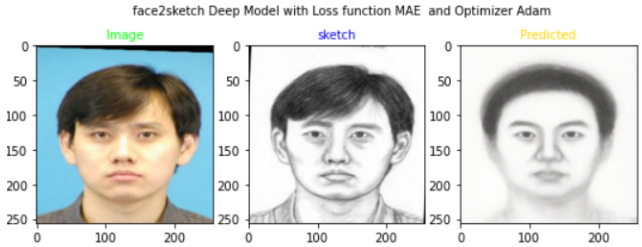

Fig. 5. Predictions of Deep Autoencoder face2sketch

A notable observation from this experiment is Fig. 6 that the deep autoencoder for sketch2face failed for Stochastic Gradient Descent Model.

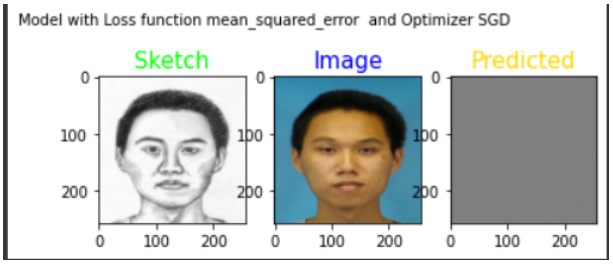

Fig. 6. SGD failed for Deep autoencoder

Fig. 7 shows the decrease in loss over training time for the Deep Autoencoder. The green line represents face2sketch model and the red line represents sketch2face model. The sketch2face records much lower training loss and doesn't change much over time indicating that model learning is very low.

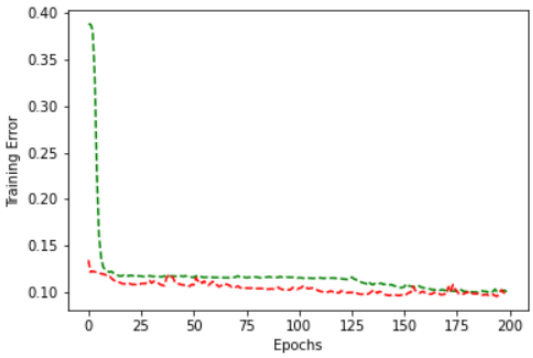

Fig. 7. Loss costs over each iteration in Deep Autoencoder

2) Convolutional Autoencoder: The summary of the Convolutional Autoencoder model used for training is presented in Fig. 8. For this model too, several optimizers and loss function were tried and the autoencoder performed best with Adam optimizer and Mean Absolute Error loss function. The sketch2face and face2sketch were trained for 200 epochs each.

As can be seen in Fig. 8, the model has around 5 million trainable parameters. The encoder model is downsampled to latent space of 32 dimensions and uses multiple convolution operations to achieve an output.

The decrease in loss for both the tasks of convolutional autoencoder can be seen in Fig. 9. We see stability in training loss in both models after around 75 epochs and the the training error stays similar after that.

Fig. 10 shows the image generated by Convolutional face2sketch model and it can be seen that the output is comparable to the test image. A major flaw in this model is the presence of noise i.e, blur elements in the image. The low resolution of the image can be attributed to the filters convolving over the image in all the different layers. But it

```
Model: "ConvolutionalAEModel"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 256, 256, 3)]     0
_________________________________________________________________
conv2d_24 (Conv2D)           (None, 256, 256, 16)      448
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 128, 128, 16)      0
_________________________________________________________________
conv2d_25 (Conv2D)           (None, 64, 64, 64)        9280
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 32, 32, 64)        0
_________________________________________________________________
conv2d_26 (Conv2D)           (None, 32, 32, 128)       73856
_________________________________________________________________
conv2d_27 (Conv2D)           (None, 32, 32, 256)       295168
_________________________________________________________________
conv2d_28 (Conv2D)           (None, 32, 32, 512)       1180160
_________________________________________________________________
conv2d_29 (Conv2D)           (None, 32, 32, 512)       2359808
_________________________________________________________________
up_sampling2d_8 (UpSampling2 (None, 64, 64, 512)       0
_________________________________________________________________
conv2d_30 (Conv2D)           (None, 64, 64, 256)       1179904
_________________________________________________________________
conv2d_31 (Conv2D)           (None, 64, 64, 128)       295040
_________________________________________________________________
up_sampling2d_9 (UpSampling2 (None, 128, 128, 128)     0
_________________________________________________________________
conv2d_32 (Conv2D)           (None, 128, 128, 64)      73792
_________________________________________________________________
up_sampling2d_10 (UpSampling (None, 256, 256, 64)      0
_________________________________________________________________
conv2d_33 (Conv2D)           (None, 256, 256, 32)      18464
_________________________________________________________________
conv2d_34 (Conv2D)           (None, 256, 256, 3)       867
=================================================================
Total params: 5,486,787
Trainable params: 5,486,787
Non-trainable params: 0
```

Fig. 8. Convolutional Autoencoder Model used for training

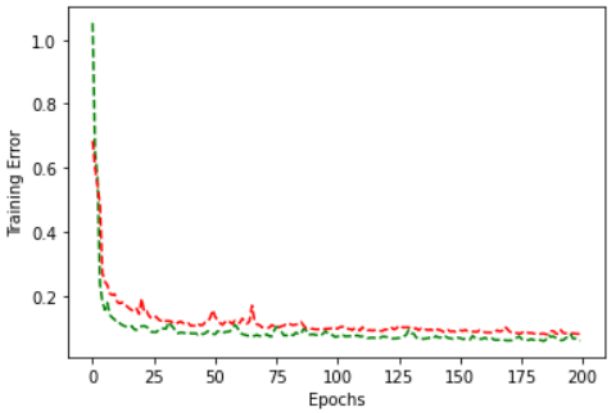

Fig. 9. Loss costs over each iteration in Convolutional Autoencoder

gives a better performance than Deep autoencoder for the same task (refer Fig 5).

Fig. 11 shows the output of the sketch2face model for convolutional autooencoder for faces of two different humans. As can be seen, the model is able to retain information about important features like hair length, face shape etc and gives a decent output. The generated images are better than the Deep autoencoder (refer Fig. 4) which learnt only limited number of features but also lacks in generating high-resolution images.

The author also attempted another convolutional model

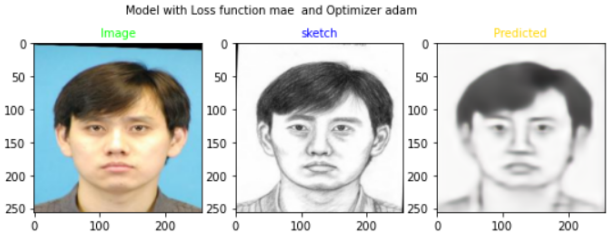

Fig. 10. Predictions of Convolutional Autoencoderface2sketch

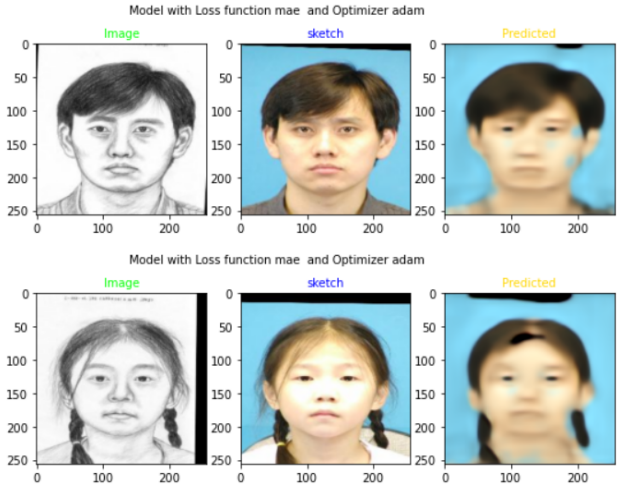

Fig. 11. Predictions of Convolutional Autoencoder sketch2face

with LeakyRelu and BatchNormalization layers to improve on existing convolutional network. Fig. 12 and Fig. 13 show the output for the same. As can be seen, the model trained over 200 epochs gives very good results for both tasks. However, one observes the decrease in brightness in this process. Using a processed image in a different color space (LAB, grayscale etc.) can be explored to reduce this discrepancy.

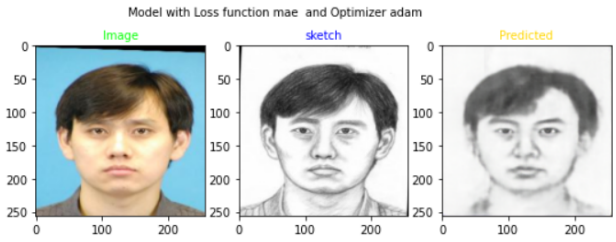

Fig. 12. Predictions of Convolutional Autoencoder with LeakyReLU and Batch Normalization sketch2face

3) Variational Autoencoder: The results of the Variational Autoencoder are not presented here since the model failed to predict images correctly for both sketch2face and face2sketch.

Table III lists out all the metrics for each of the model trained in this experiment. The performances of Deep, Convolutional (Conv) and Convolutional encoder with leakyReLU and batch normalization layer are listed for each of tasks - sketch2face (s2f) and face2sketch (f2s).

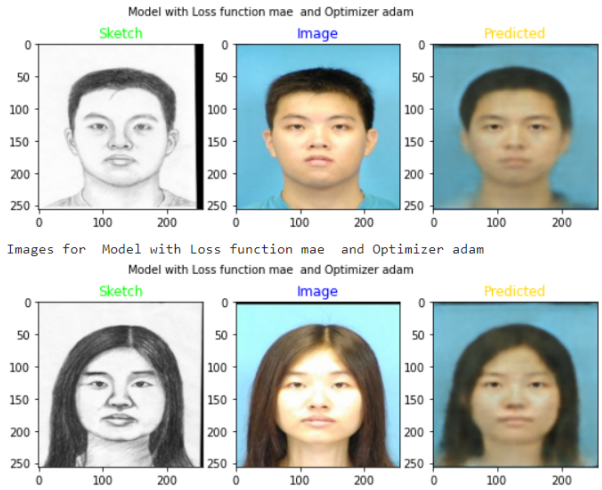

Fig. 13. Predictions of Convolutional Autoencoder with LeakyReLU and Batch Normalization face2sketch

Model Name	Loss	Accuracy	MAE	PSNR	SSIM
Deep f2s	0.10	50.8	0.11	13.63	0.57
Deep s2f	0.172	83.1	0.1968	11.66	0.3443
Conv f2s	0.0727	71.7	0.0618	19.38	0.6715
Conv s2f	0.085	89.5	0.1396	12.642	0.5773
Conv with Relu f2s	0.0814	84.6	0.079	17.53	0.632
Conv with Relu s2f	0.146	84.1	0.186	13.51	0.637

TABLE III
SIMILARITY MEASURE INDEXES FOR ALL MODELS

Variational autoencoder (VAE) model is removed from the results due to incorrect training and output images.

V. Discussion

From above experiments, the study concludes that convolutional autoencoders are easier to train and give good results for image encoding and decoding tasks. Addition of Batch Normalization and Relu activation function improves their performance slightly but it can be made much better with extensive training and hyperparameter tuning.

The author recognises several other approaches to accomplish the project tasks. The most popular ones are- using multiple GAN models (such as CycleGAN, pix2pix etc) and Autoencoders with higher complexities (such as conditional variational autoencoder etc).

Several image pre-processing techniques before model training can be used before model training to develop faster models. Converting images from RGB (or BGR) format to LAB format (Luminance and two color channels) or Gray scale can lower the count of training parameters greatly.

The use of high-contrast images in training sketch2face model is a topic that has not been explored much in the recent years and is a research topic that the author was interested in.

Use of sobel edge detector to find edges from image and inverting the image to generate a successful black and white sketch image was ideated but not implemented successfully. The author would like to training a generative model on sobel edges to generate RGB images.

All the models trained in the experiment take a global approach to this problem by working on the entire image instead of some parts of it i.e, local features in the image. The latter allows a more granular approach to the problem as shown in DeepDrawing.

VI. Conclusion

This study has been focused on understanding deep autoencoders and trying out their implementations in the context of generating sketch images from actual photos and generating photos from plain sketch images. Various optimizers and loss functions were tried to seek the combination that gives the best performance. The execution of models was limited due to high training time for each model and trying different combinations of hyperparameters on the same model is computationally exhaustive.

Link to the implementation can be found at - google colab

Acknowledgment

This author would like to thank the following authors for the github code for some of the autoencoders developed in this report - roatienza, paperspace, Ali Abdelaal, Odegeasslbc. Several blogs require a special mention for multiple informational posts - machinelearningknowledge, machinelearningmastery, UvA DL Notebooks, Jeremy Jordan. Many thanks to the brains behind helpful computer vision libraries like OpenCV, Keras, scipy, skimage etc.

References

[1] Stacked multichannel autoencoder – an efficient way of learning from synthetic data. 77(20):26563–26580, 2018.

[2] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495, 2017.

[3] SE Brennan. Caricature generator [ms. thesis]. *MA: MIT*, 1982.

[4] Darren Burke and Danielle Sulikowski. The evolution of holistic processing of faces. *Frontiers in Psychology*, 4, 2013.

[5] Tissa Chandesa and Michael Hartley. Automated sketcher using edge detection techniques. *International journal of computers applications*, 32(4):404–411, 2010.

[6] Shu-Yu Chen, Wanchao Su, Lin Gao, Shihong Xia, and Hongbo Fu. Deepfacedrawing: deep generation of face images from sketches. *ACM transactions on graphics*, 39(4):72:1–72:16, 2020.

[7] Xing Di and Vishal M Patel. Face synthesis from visual attributes via sketch using conditional vaes and gans. 2017.

[8] Deng-Ping Fan, ShengChuan Zhang, Yu-Huan Wu, Ming-Ming Cheng, Bo Ren, Rongrong Ji, and Paul L Rosin. Face sketch synthesis style similarity:a new structure co-occurrence texture measure, 2018.

[9] Liang Fan. *Face sketch recognition using deep learning*. PhD thesis, Cardiff University, 2020.

[10] Davide Giordano. 7 tips to choose the best optimizer, 2020.

[11] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[12] Jeremy Jordan. Variational autoencoders. https://www.jeremyjordan.me/variational-autoencoders/.

[13] Ibrahem Kandel, Mauro Castelli, and Aleš Popovič. Comparative study of first order optimizers for image classification using convolutional neural networks on histopathology images. *Journal of imaging*, 6(9):92, 2020.

[14] Agustinus Kristiade. Agustinus kristiadi's blog, 2016.

[15] Sachit Menon, Alexandru Damian, Shijia Hu, Nikhil Ravi, and Cynthia Rudin. Pulse: Self-supervised photo upsampling via latent space exploration of generative models. 2020.

[16] Paperspaceblog. How to build variational autoencoders.

[17] Chitwan Saharia, Jonathan Ho, William Chan, Tim Salimans, David J Fleet, and Mohammad Norouzi. Image super-resolution via iterative refinement. 2021.

[18] Xiaogang Wang and Xiaoou Tang. Face photo-sketch synthesis and recognition. *IEEE transactions on pattern analysis and machine intelligence*, 31(11):1955–1967, 2009.

[19] Xiaogang Wang and Xiaoou Tang. Face photo-sketch synthesis and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(11):1955–1967, 2009.

[20] Zhou Wang, A.C Bovik, H.R Sheikh, and E.P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[21] Mingjin Zhang, Jie Li, Nannan Wang, and Xinbo Gao. Compositional model-based sketch generator in facial entertainment. *IEEE transactions on cybernetics*, 48(3):904–915, 2018.

[22] Hao Zhou, Zhanghui Kuang, and K. K Wong. Markov weight fields for face sketch synthesis. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1091–1097. IEEE, 2012.