The BRITISH UNIVERSITY IN EGYPT

Deep Learning Phase 2 Report

Autoencoders and Variational Autoencoders

Group 12

ID	Name	Contribution
192895	Hanin Monir Ismail	VAE and loss functions
206562	Jacinta Samir	Pre-processing and Denoising using AE
203800	Youssef Ayman	VAE and loss functions

Table of Contents

Application 3
Dataset 3
Proposed Methodology 3
https://colab.research.google.com/drive/1blr60DlQ-_KRzxpZcguZNKKn7X0eKcD5?usp=sharing 3
Pre-processing 3
Denoising using an AE 5
Metrics 7
Hyperparameters 7
Conclusion 9
References 10

Application

We aim to generate faces with certain added features, such as making them wear glasses, using VAEs. This will be done by deriving the latent vector of a given feature (using latent space arithmetic).

Dataset

The dataset we will use is CelebA [1], which comprises over 200,000 images of over 10,000 various celebrities. Each image is tagged with attribute annotations such as ‘Smiling,’ ‘Eyeglasses,’ ‘Moustache,’ ‘Wearing hat,’ and many more (40 different labelled attributes). One of these attributes will be used in the VAE to generate new images with said attribute. For instance, a ‘Smile Vector’ has previously been deduced and used to, given an input face, output a smiling version of the face [2]. In a similar manner, we will discover a latent attribute vector, and use it to generate new images that apply the attribute onto the inputted faces.

Proposed Methodology

Colab notebook:

https://colab.research.google.com/drive/1blr60DlQ-_KRzxpZcguZNKKn7X0eKcD5?usp=sharing

Pre-processing

Initially, only the first 10,000 images of the dataset were taken and uploaded to the drive for use. A sample of the images can be seen in Fig. 1:

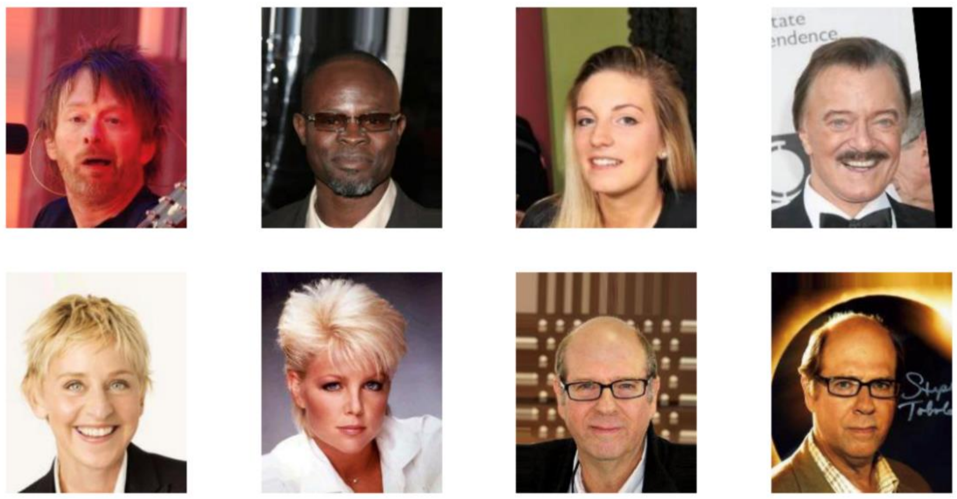

Fig. 1: Sample of images from the CelebA dataset.

As seen, all images in the dataset were cropped and aligned to the faces of the celebrities. The sizes of all images were checked to ensure they are all equal and were all found to be 178x218.

To be able to train and test the Denoting Autoencoder (DAE), another dataset of noisy images is required. Thus, random Gaussian noise was added to the original dataset to create an additional noisy dataset. Fig. 2 shows a sample of images from said noisy dataset, and Fig. 4 depicts a comparison of a clean and noisy image.

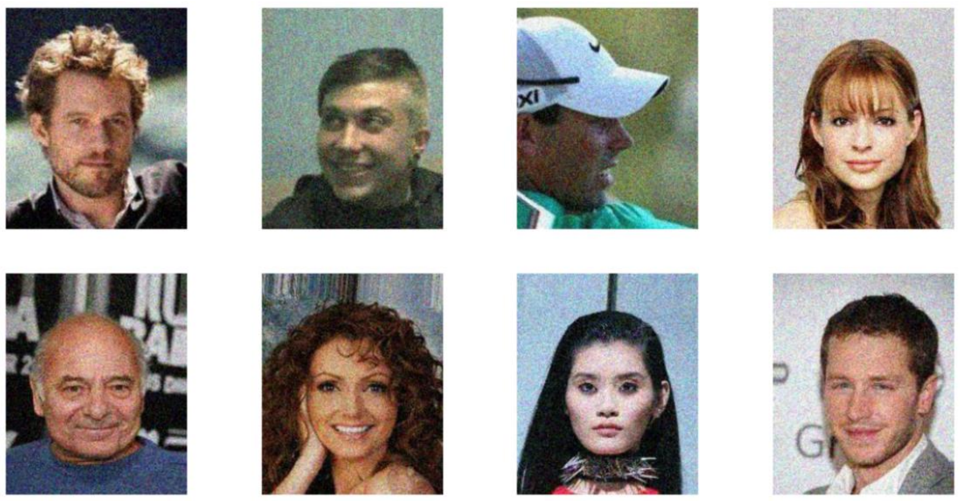
Fig 2. Sample of images from the Noisy dataset.

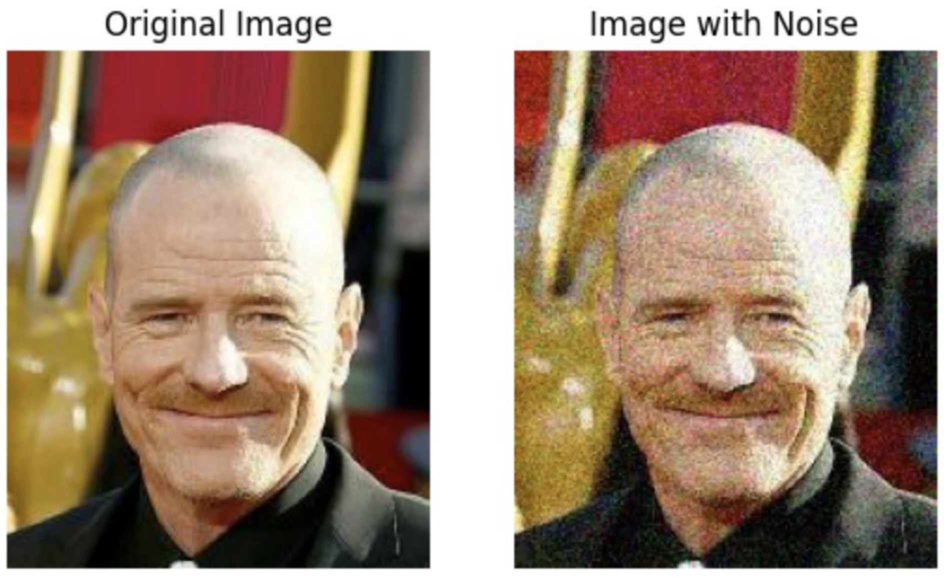

Fig. 3: A sample image before and after adding random Gaussian noise.

Since the sizes of the images were 178x218, the reduction of the dimensions within the DAE would result in odd numbers, which would then be upsampled to a slightly larger size of 180x220. The input and output are required to be of the same size, and thus the images were padded with a single-pixel border.

Next, both the original and noisy datasets were saved into an array format, in order to feed both simultaneously into the model (with the clean images acting as target labels). Additionally, the pixel values were normalized to be within 0 and 1. Due to the normalization increasing the size of the arrays (as conversion to float was needed), only 1000 images were used, due to limitations in computational resources (**see note below*). Both arrays were then split into training and testing sets, 60% and 40% respectively.

Denoising using an AE.

A DAE was then constructed as follows: an input layer, two convolutional (2D) layers (to down sample the dimensions into a latent representation), two transposed convolutional layers (to up sample back to the original input size), and finally an output layer). This creates a bottleneck between the encoder (Conv2D layers) and the decoder (Conv2DTranspose layers) in which the data is reduced to from (220, 180, 3) to (55, 45, 16) dimensions. All layers used ReLU activation, with the exception of the final output layer, which used sigmoid. Due to the sigmoid layer, the appropriate loss function used was binary cross-entropy, a common loss function used in autoencoders (to minimize the reconstruction loss). The DAE was trained over 10 epochs, using a batch size of 32.

The DAE achieved a testing loss of 0.674, and its denoised output can be found in Fig. 4.

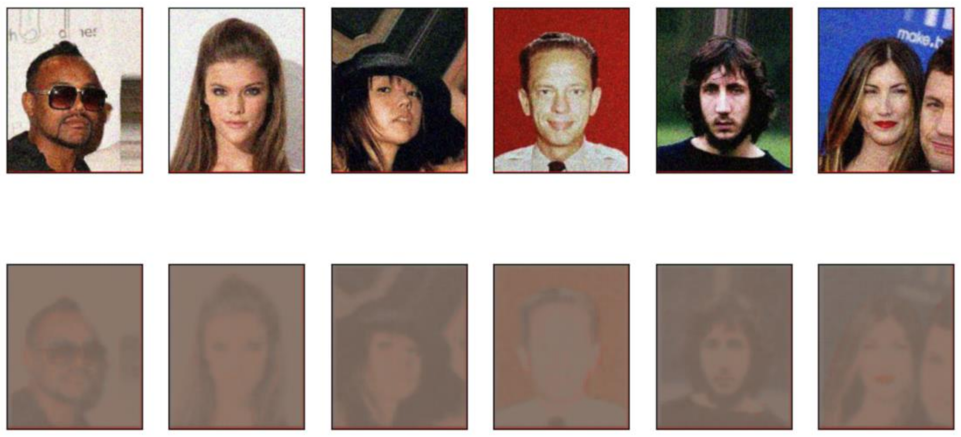

Fig. 4: DAE output

It is evident that the DAE model was not very successful in achieving its goal to Denoise the images, as the outputs have a significant brown cast, and lack detail. The model could be improved by advancing its architecture, and likely needed to be trained for more epochs, and on a larger sample of the dataset. However, due to the limitations of the available resources, doing so was unfeasible.

**Note*: The primary DAE attempted implemented a normalization layer directly onto the input and was trained on all 10k images for 3 epochs. However, not only did it take 45 minutes to train, but also failed in learning, as it resulted in a test loss of –76805200.0 and predicted entirely blank white images. This was because the loss function (BCE) compares the predicted probabilities (which were normalized in the rescaling input layer) to the actual ground-truth images (which did not undergo the normalization process). For this reason, a sample of both the original and noisy were normalized prior to feeding the model.

Latent Arithmetic using VAE.

The CelebA dataset contains binary attribute data for each image, describing what the image contains. For instance, if the person in the image is smiling, then the smiling attribute of that image will be set to 1. We aim to build a model similar to that of Hou et al [3] that adds smiles to faces using a VAE by calculating a smile vector for an image and multiplying that vector by a “smile factor” then adding the resultant in the latent space embedding. Finally, the result is fed to the decoder.

For this model, the CelebA dataset was loaded from TensorFlow so that we can use all 200,000 images and their attributes. The images were resized to 128x128, and the pixel values normalized. After splitting, the training set had 162770 images and the testing set had 19962 images, a ratio of 80 to 20. The VAE model was constructed with an encoder that had a input size equal to the image size (128x128x3), the latent space that was initial of 100 dimensions and a decoder, using the KLD and the reconstruction loss.

Because of the limit processing power the code in Colab failed to train and the dataset wasn’t properly loaded, so there aren’t any discrete results for the model.

After training the model, to add smiles to the face, a sample from the test data was taken and fed to the model’s encoder, then the results was added to the product of the chosen smile factor and a calculated smile vector. Finally, the resultant, was fed to the decoder.

Metrics

We use Reconstruction loss and KL divergence to evaluate our models since that they are important metrics for evaluating autoencoders and variational autoencoders (VAEs). Moreover, tracking them over the course of training can provide valuable insights into models' performances. In addition to using TensorBoard, an effective tool for visualizing and tracking these metrics, it allows for real-time monitoring of model training and evaluation metrics. Finally, learning curves that plot the change in these metrics over time, can help identify problems such as overfitting, and can guide decisions regarding model architecture and hyperparameter tuning.

Because the VAE model wasn't trained due to problems with the processing power, it wasn't evaluated but the code of the metrics was implemented.

Figure 5 illustrates the learning curve of the trained DAE model. It shows that the model wasn't done learning because it was still declining and didn't reach a plateau.

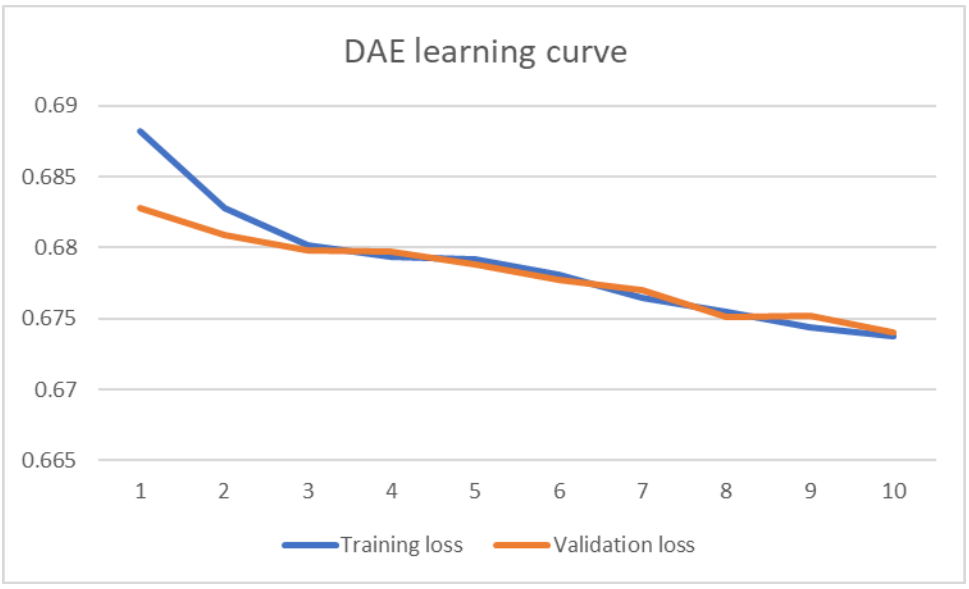

Fig. 5: DAE learning curve plotting epoch against loss.

Hyperparameters

The hyperparameters we tuned were:

- Learning rate
- Batch size
- Latent space dimension
- Number of epochs
- Number of filters in each layer
- Kernal size of each layer
- Number of layers of the encoder
- Number of layers of the decoder
- Encoder strides
- Decoders strides
- Activation function

Hyperparameter values for the DAE:

- Input size: (220x180x3)
- Number of layers of the encoder: 2 convolutional layers
- Number of layers of the decoder: 2 convolutional transpose layers and 1 output layer
- Number of filters in each layer
    - Encoder layer 1: 32
    - Encoder layer 2: 16
    - Decoder layer 1: 16
    - Decoder layer 2: 32
- Activation function: Relu for the middle layers and sigmoid for the output layer
- Encoder strides: 2
- Decoders strides: 2
- Learning rate: 0.01
- Batch size: 32
- Epochs: 10

Hyperparameter values for the VAE:

- Input size: (128x128x3)
- Latent dimension: 100, the larger the dimension of the latent space the better the information will be captured.
- Number of layers of the encoder: 2 convolutional layers
- Number of layers of the decoder: 2 convolutional transpose layers and 1 output layer
- Number of filters in each layer
    - Encoder layer 1: 32
    - Encoder layer 2: 64
    - Decoder layer 1: 64
    - Decoder layer 2: 32
- Activation function: Relu for the middle layers and sigmoid for the output layer
- Encoder strides: 2
- Decoders strides: 4
- Learning rate: 0.1

- Batch size: 64
- Epochs: 10

Conclusion

In summary, our report outlines our approach to augmenting facial features, such as adding smiles, in generated faces using Variational Autoencoders (VAEs) and latent space arithmetic. We utilized a large dataset of celebrity images, called CelebA, which includes attribute annotations like 'Smiling,' 'Eyeglasses,' 'Moustache,' etc. Our methodology involved pre-processing the dataset by resizing and normalizing the images and creating a noisy dataset by adding random Gaussian noise. We then trained a Denoising Autoencoder (DAE) to denoise the images but faced challenges due to limited computational resources. Nonetheless, we gained valuable insights into the importance of model architecture and training duration. Next, we implemented a VAE to perform latent arithmetic and generate images with the desired attribute. Inspired by the work of Hou et al. [3], we calculated a smile vector for an image, multiplied it by a "smile factor," and added the result to the latent space embedding before decoding. However, we were not able to produce discrete measurable results due to the limited computing resources. Further experimentation with different model architectures, training durations, better computing resources and larger datasets could lead to better results in denoising and attribute-based image generation. Our work contributes to the field of image generation with added features using VAEs and latent space arithmetic. We hope that our findings inspire further research and advancements in this area, with potential applications in domains such as entertainment, virtual reality, and computer vision.

References

[1] Liu, Ziwei & Luo, Ping & Wang, Xiaogang & Tang, Xiaoou, “Deep Learning Face Attributes in the Wild,” 2014, doi: 10.1109/ICCV.2015.425.

[2] T. White, “Sampling Generative Networks”. Open Access Te Herenga Waka-Victoria University of Wellington, Dec. 2016, doi: 10.26686/wgtn.12585362.v1.

[3] X. Hou, L. Shen, K. Sun and G. Qiu, "Deep Feature Consistent Variational Autoencoder," 2017 IEEE Winter Conference on Applications of Computer Vision (WACV), Santa Rosa, CA, USA, 2017, pp. 1133-1141, doi: 10.1109/WACV.2017.131.