Image generation with Variational Autoencoders (VAE)

Kristian Perriu [505571]

Chibuzor John Amadi [502623]

University of Pavia, Pavia

University of Milano-Statale, Milano

University of Milano-Bicocca, Bicocca

1. Abstract

Variational Autoencoders (VAE) [1] have emerged as one of the most popular approaches used for unsupervised learning of complicated distributions. They are appealing as they are built on top of standard neural networks [2] and can be trained using gradient descent [3]. They have shown great performance in generating different type of complex data such as handwritten digits, faces, house numbers and CIFAR images [4].

2. Introduction

2.1 Dataset

The dataset used in this report is the CelebA dataset [5] which is a large-scale face attributes dataset with more than 200K celebrity images, each with 40 attribute annotations. The images in this dataset cover large pose variations and background clutter. CelebA has large diversities, large quantities, and rich annotations, including 10.177 number of identities, 202.599 number of face images, and 5 landmark locations, 40 binary attributes annotations per image. Please find more information here.

2.2 Data Preprocessing

For this task, the preprocessing was kept relatively simple. The images were at first resized to 64 by 48 pixels and a random horizontal flip [6] was applied to the dataset. The images were then turned to tensors [7] and then normalized.

3. Model architecture

The model itself contains two main modules which are the encoder and the decoder. They are both convolutional neural networks and they are symmetrical to each other. The convolutional layers used are 2d with a stride and kernel size that changes every layer while the padding is kept at 1. And instead of using batch normalization [8], instance normalization [9] is applied because batch version normalizes all images across the batch and spatial locations and instance version normalizes each element of the batch independently, across spatial locations only thus making each individual image distribution look Gaussian [10], but not jointly.

3.1 Encoder module

The encoder module consists of multiple convolutional layers [21] connected to each other. The model itself keeps expanding from layer to layer and the parameters for each layer change as well. The module starts at 32 channels and ends up at 256. Such thing will be displayed below on the table representing the architecture of the model. The encoder also includes two fully connected layers [11] which are responsible for generating the

1 and the log variance [12]. These layers input is defined by the multiplication of the width and the height of the model divided by 16 and the output will be the number of latent features [13].

The input is passed through the convolutional layer, then through the instance normalization layer and then through a ReLU [14] activation function. Such process is repeated 8 times for our model and then the output of these layers is flattened, and passed through the said fully connected layers mentioned above.

Table 1, Architecture of the encoder module.

Convolutional layer [in_channels=3, out_channels=32, kernel_size=4, stride=2, padding=1]
Instance normalization [num_features =32]
ReLU
Convolutional layer [in_channels=32, out_channels=32, kernel_size=3, stride=1, padding=1]
Instance normalization [num_features =32]
ReLU
Convolutional layer [in_channels=32, out_channels=64, kernel_size=4, stride=2, padding=1]
Instance normalization [num_features =64]
ReLU
Convolutional layer [in_channels=64, out_channels=64, kernel_size=3, stride=1, padding=1]
Instance normalization [num_features =64]
ReLU
Convolutional layer [in_channels=64, out_channels=128, kernel_size=4, stride=2, padding=1]
Instance normalization [num_features =128]
ReLU
Convolutional layer [in_channels=128, out_channels=128, kernel_size=3, stride=1, padding=1]
Instance normalization [num_features =128]
ReLU
Convolutional layer [in_channels=128, out_channels=256, kernel_size=4, stride=2, padding=1]
Instance normalization [num_features =256]
ReLU
Convolutional layer [in_channels=256, out_channels=256, kernel_size=3, stride=1, padding=1]
Instance normalization [num_features =256]
ReLU
Flattening layer
Fully connected layer for μ, Fully connected layer for log var

3.2 Decoder module

The decoder module, as for the architecture is similar to the encoder module although they have some important differences. At first the input is passed through a fully connected layer with the same dimensions of the fully connected layers in the encoder module and then is reshaped. Instead of applying a convolutional layer, the decoder module implements a transpose convolutional layer [15] which is also known as a fractionally-strided convolution or a deconvolution. Such layer increases the spatial dimensions of the input data thus making it fitting for image generation. This module takes the output of the final convolution layer and passes it through a tanh function [16] so the output values can be between 1 and -1 which is typical for image generation tasks.

Table 2, Architecture of the decoder module.

Fully connected layer
Reshape input
Transpose convolutional layer [in_channels=256, out_channels=256, kernel_size=3, stride=1, padding=1]
Instance normalization layer [num_features=256]
ReLU
Transpose convolutional layer [in_channels=256, out_channels=128, kernel_size=4, stride=1, padding=1]
Instance normalization layer [num_features=128]
ReLU
Transpose convolutional layer [in_channels=128, out_channels=128, kernel_size=3, stride=1, padding=1]
Instance normalization layer [num_features=128]
ReLU
Transpose convolutional layer [in_channels=128, out_channels=64, kernel_size=4, stride=2, padding=1]
Instance normalization layer [num_features=64]
ReLU
Transpose convolutional layer [in_channels=64, out_channels=64, kernel_size=3, stride=1, padding=1]
Instance normalization layer [num_features=64]
ReLU
Transpose convolutional layer [in_channels=64, out_channels=32, kernel_size=4, stride=2, padding=1]
Instance normalization layer [num_features=32]
ReLU
Transpose convolutional layer [in_channels=32, out_channels=32, kernel_size=3, stride=1, padding=1]
Instance normalization layer [num_features=32]
ReLU
Transpose convolutional layer [in_channels=32, out_channels=3, kernel_size=4, stride=2, padding=1]
Tanh function

3.3 Training loop

The loss function applied to this model is the ELBO loss function, also called the Evidence Lower Bound [17]:

$$ELBO = Eq[\log q(z|x) - log\ p(z)] - Eq[log\ p(x|z)]$$

Where:

- $q(z \mid x)$ is the variational posterior, an approximation of the true posterior p(z | x).
- p(x | z) is the likelihood, representing the generative process.
- p(z) is the prior distribution over the latent variables z .
- D_{KL} denotes the Kullback-Leibler divergence.

This function serves as a lower bound to the marginal likelihood p(x). Maximizing such function is equivalent to minimizing the difference between the approximate posterior q(z|x) and the true posterior p(z|x), while also maximizing the likelihood of the data.

4. Evaluation

The model was modified in order to establish how a change in parameters will affect the model's ability to generate faces and perform interpolation.

4.1 Experimental details

The used optimizer for such task as the Adam [18] optimizer with a learning rate [19] of 0.001. These two parameters were not experimented with any further. Other parameters that were subject to experimentation were the parameter LAMBDA which is responsible for weighting the contribution of the Kullback-Leibler (KL) [20] divergence term in the overall loss function. The other parameter is the latent space which is responsible for encoding the information about the input data in a compressed and continuous form. The latent space or the so-called *z* variable was also made closer and further away from the center. Due to the KL divergence, the *z* variable is inherently in the center but to make it even closer to it, a change in the KL divergence was made to increase its effect. This can be simply done by adding 1 to such function.

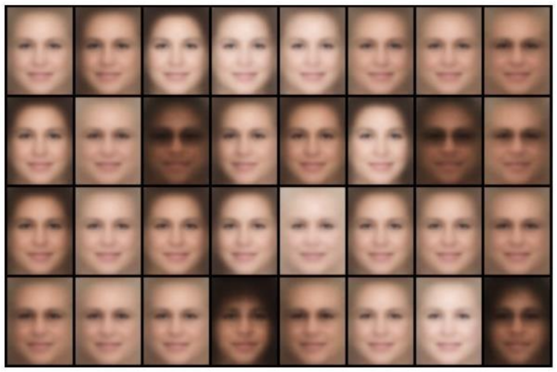
The specific values used for LAMBDA were 10^{-3}, 10^{-4}, $10^{-5}10^{-6}$and for the latent space 32, 64 and 128. The model that performed the best when LAMBDA was changed was also subject to a change of the latent space.

The first batch of pictures is generated by a model with a latent space of 64. The other pictures generated by the model with other latent space sizes are displayed further below.

The model will be based on the quality of the faces it generates and its ability to perform interpolation.

Figure 1, Face generation for lambda=10^{-3}

Figure 2, Face generation for lambda= 10^{-4}



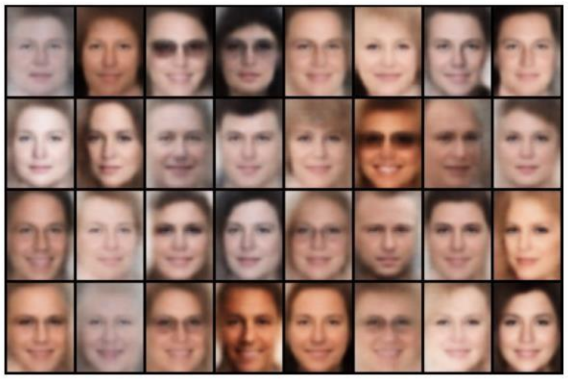
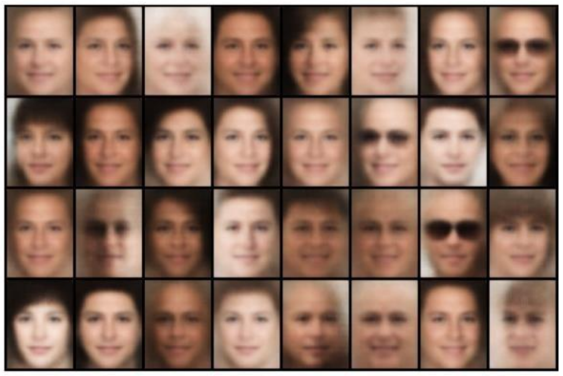


Figure 3, Face generation for lambda= 10^{-5}

Figure 4, Face generation for lambda=10^{-6}



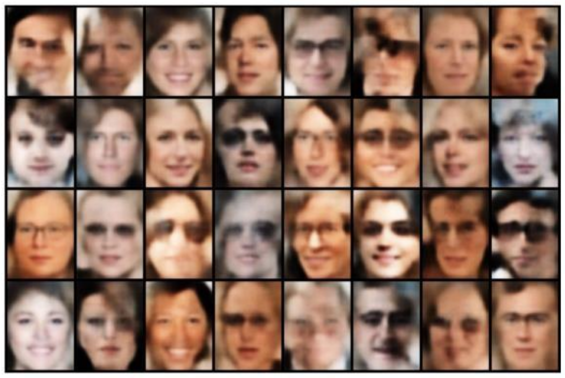
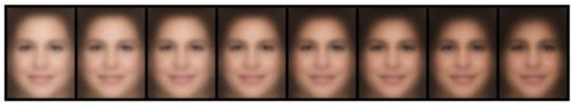


Figure 5, Interpolation for lambda=10^{-3}

Figure 6, Interpolation for lambda=10^{-4}



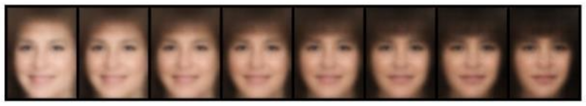
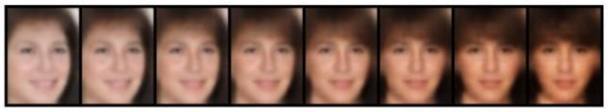
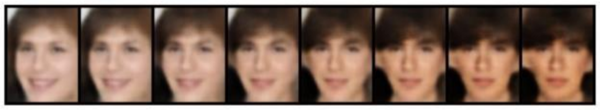


Figure 7, Interpolation for lambda=10^{-5}

Figure 8, Interpolation for lambda=10^{-6}

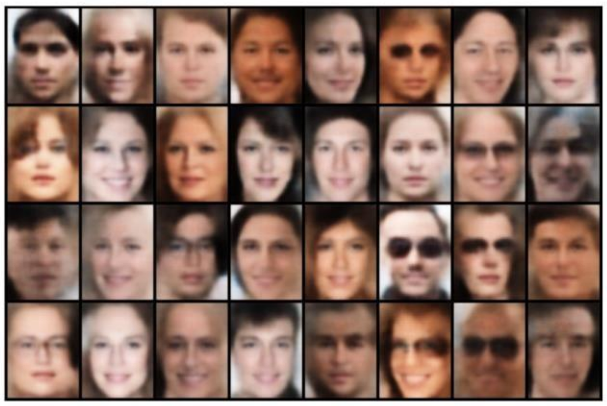




Since lambda=10^{-5} had the best performance, it will be subject to a change also in the size of the latent space. Such thing will be proved below in the result analysis.

Figure 9, Face generation for latent space=32

Figure 10, Face generation for latent space=128



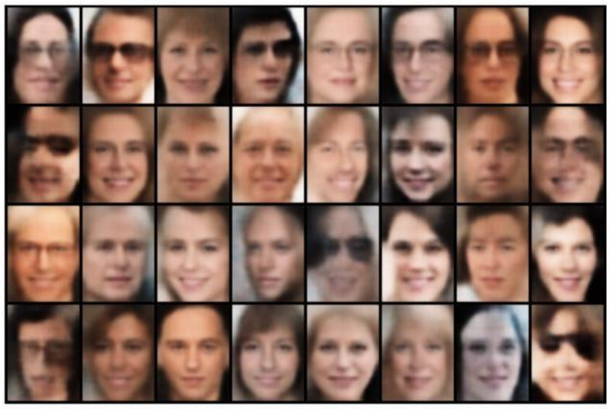
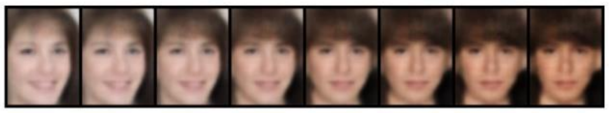
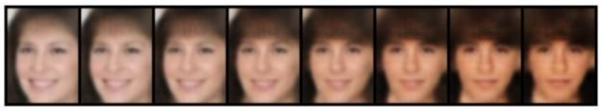


Figure 11, Interpolation for latent space=32

Figure 12, Interpolation for latent space=128





Further experiments were carried with the best model. An attempt to make the variable closer to the center was made and the results are below.

Figure 13, Face generation when z was closer to the center.

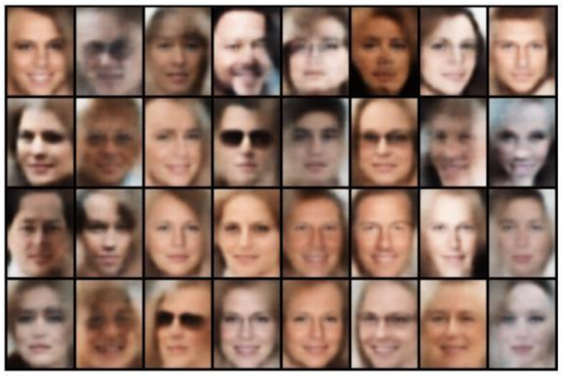
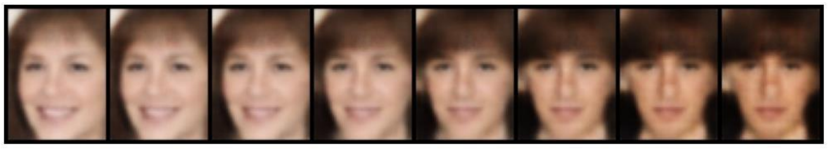


Figure 14, Interpolation when z was closer to the center



4.2 Result analysis

When lambda is increased at 10^{-3} the model tends to struggle with minimizing the loss. As seen, the faces generated by the model this time, tend to be very symetrical and only the front of the face is generated, that is, the main features such as the eyes, the nose and the mouth. The side of the face seems to be classified as an unimportant feature by such model and there is hardly any change even when interpolation is performed. See figures 1 and 5.

When lambda was kept at 10^{-4} the model resulted in more detailed generated faces. Even features we did not see before in the case before started to appear. Some that we can mention are the jaw and the angle given to some faces and these features are also decently defined with respect to the first model. The interpolation ability has drastically improved as we can see a change of skin tone and head angle along the displayed pictures. See figures 2 and 6.

Lambda was further decreased to 10^{-5} and we can see that the model now has way more variability in the faces generated and the model now generates features such as glasses, the side of the face, and the jawline more consistently than before. Such features before were considered irrelevant when lambda was too high. A faster decrease in the loss was also noticed. Now even when performing interpolation, the transition between the faces is decently smooth and also the final face has more details compared to other models with a higher lambda. See figures 3 and 7.

When lambda was decreased even further to 10^{-6} the faces started to deform and features such as the jaw which was clearly constructed by the models before, now is exaggerated and deformed. Such deformities are also noticed in the upper part of the head where the side forehead is. Such phenomena is also seen even when glasses are present in the picture as that region is deformed as well. Regardless of these results, the model has no impairment when interpolation was performed as it can smoothy transition between two faces. See figures 4 and 8.

The model that performed the best was the one with lambda 10^{-5} so this model, as mentioned above will have a change in its latent space as well.

In both cases where the latent space was changed, we have similar results in the face generation as when the lambda was very low. Although in the model which had a latent space of 128, when there are no confusing features such as the glasses the model performs fairly well and constructs very detailed faces but when glasses are introduced it results in a lot of deformities. Such phenomena can be seen even when the latent space was small. In both interpolation instances we still see no impairment on the models' ability to perform such operation. For the model with a latent space of 32 see figures 9 and 11 and for the model with a latent space of 128 see figures 10 and 12.

After continuing with the model with lambda and latent space 10^{-5} and 64 respectively, an attempt to make z more centered to the gaussian distribution was made, as mentioned above, to see how the model would behave. As always, the interpolation ability remains intact but we can see a similar phenomenon that happens when the latent space is too big or too small, that is, the model makes important features more distinct and results in deformities where the unimportant features are present. This mainly happens when the generated pictures have a very warm skin tone. A different result is seen when the generated pictures have light skin tones as there are more deformities.

5. Results and conclusion

As seen the more the lambda decreased the better the model performed but this was seen until it reached 10^{-5} as when it got too low the model performed very poorly. We can assume there has to be an optimal lambda for the model to have maximal performance. The same phenomenon was noticed when the latent space was subject to change. When it was too large, although it generated very detailed faces it failed to construct faces properly it failed to do so when different features such as glasses were introduced. This feature specifically is not considered very important by all the models and thus the instances where it was present were not the best generated. We can assume that a change in the latent space further improves the model's ability to produce important features such as the face nose and eyes but it also makes features that could be considered unimportant such as the glasses even more unimportant and thus resulting in a lot of deformities.

It is safe to assume that after a low enough lambda the model's ability to perform interpolation remains impaired and very consistent as such thing was seen with 7 different models. The latent space also had no significant effect on interpolation as well.

When the variable z is too close to the center, it fails to properly generate faces with very white skin and hair, specifically on female faces. This could be to probably imbalance in data or even the probably not very distinct facial features of such faces that might be present in the dataset as this phenomenon is not seen in warmer skin tones.

Better results could be seen with a more rigorous experimentation with parameter combinations but this was not achievable due to restricted computational resources. Such results could also be achieved with a better modification of the loss function.

6. References

[1] Variational Autoencoders https://arxiv.org/pdf/1606.05908

[2] Neural networks https://en.wikipedia.org/wiki/Neural_network_(machine_learning)

[3] Gradient descent https://en.wikipedia.org/wiki/Gradient_descent

[4] CIFAR images https://en.wikipedia.org/wiki/CIFAR-10

[5] CelebA dataset https://mmlab.ie.cuhk.edu.hk/projects/CelebA.html

[6, 7, 8, 9, 11, 14, 15, 18, 19, 21] https://pytorch.org/

[10] Gaussian https://en.wikipedia.org/wiki/Gaussian_process

[12] Log variance https://en.wikipedia.org/wiki/Log-normal_distribution

[13] Latent features https://en.wikipedia.org/wiki/Latent_space

[16]	tanh function	https://en.wikipedia.org/wiki/Hyperbolic_functions
[17]	Evidence Lower Bound	https://en.wikipedia.org/wiki/Evidence_lower_bound
[20]	KL divergence	https://en.wikipedia.org/wiki/Kullback%E2%80%93Leibler_divergence