Variational Autoencoders (VAE)

Methodology Report

Carolina Macedo, Fengyi Chen, Emilie Guido, Nana Maglakelidze, and Gaëtan Drouet

Ecole Nationale des Chartes-PSL
Digital Humanities
June 2024

Submitted to fulfill the requirements
for the Master's Degree in Digital Humanities

Contents

0.1 Presentation of the project

The project uses a Variational Autoencoder (VAE) to generate images of the letter E from the pre-existing image dataset which are crops of the letter *epsilon* on *papyri*. We used images of the letter *epsilon* to begin with. The approach is based on the use of TensorFlow and Keras to build and train the VAE model. The images are pre-processed by converting the colours, resizing to a standard size of 64x64 pixels, and normalising the pixel values between 0 and 1, which is fairly ordinary pre-processing.

0.2 Method

To put it as simply as possible, the VAE consists of two main parts: an "encoder" and a "decoder". The encoder takes an image of the letter E as input and transforms it into a set of parameters that describe an abstract space called "latent space". This latent space captures the essential aspects of the image. A stochastic sampler takes these parameters and randomly generates points in this latent space. This allows the model to explore different possible variations of the original image. The decoder takes these points in latent space and uses them to reconstruct a new version of the image of the letter E. This reconstruction is an attempt by the model to generate an image similar to the original based on its knowledge of latent space.

Once the VAE has been trained on a set of images of the letter E, it becomes capable of generating new images. To generate a new image, we simply give it a random latent vector as input to the decoder. This latent vector represents a unique combination of attributes that the model has learned during training. The decoder uses this latent vector to produce a new image of the letter E, which may be different from those in the training dataset but retains the general characteristics of the letter E.

To assess how well our model performs, we examine the generated images. We visually compare these images with those in the original dataset to see if the VAE manages to generate recognisable and consistent images of the letter E.

0.3 Attempts to Adjust Hyperparameters

- Our first attempt led us to train the model over 30 epochs with a batch of 25 and a latent space of 200. The choice of these hyperparameters was exploratory. We generated just 10 images to see what the results were. The images generated do represent an *epsilon* but they clearly lack sharpness.

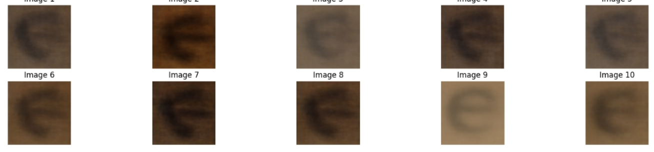
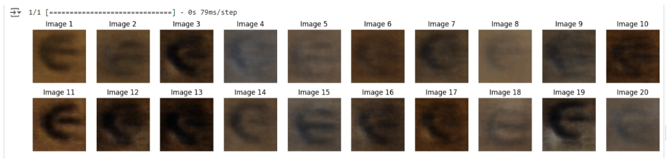


Figure 1: epochs=30, batch-size=25, latentsdim = 200

- After 30 training epochs with a batch size of 35, our VAE model showed a progressive reduction in loss, from 8240.0947 to 7700.4570. To enrich the evaluation of the model's generation diversity, we increased the number of images generated to 20 while reducing the latent space to 100 to study its impact. The improvement in the representation of the letter *epsilon* compared with previous results remains modest; the characters generated remain generally unclear.



```
1/1 [==============================] - 0s 79ms/step
```

Figure 2: epochs=30, batch-size=35, latentsdim = 100

- The number of epochs was therefore increased, as was the batch size: 100 epochs and a batch of 65. We again observed a continuous decrease in the loss, which is positive since it indicates that the model is continuing to learn and adjust. However, despite this quantitative improvement, the qualitative results of the images generated showed unexpected degradation. The images generated show a noticeable change in colour, with a marked tendency towards blue tones. This may be due to an imbalance in the reconstruction of the colour channels by the decoder.

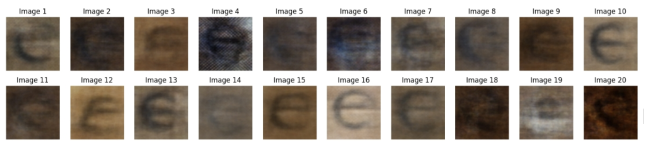
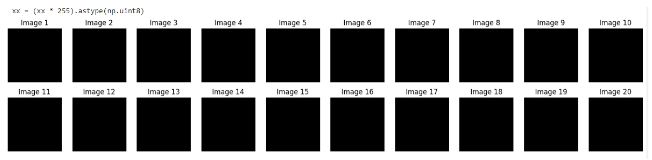


Figure 3: epochs=100, batch-size=65, latentsdim = 100

- One solution we tried was not to focus on the hyperparameters but instead to change the image resolution to 128x128 rather than 64x64.After adjusting the resolution of the input images to 128x128 pixels, we observed that the images generated all appeared black. This outcome could be attributed to several factors. Firstly, it is possible that the model has not yet learned to handle the characteristics of higher resolution images with the same number of epochs and the same architecture. Secondly, the increased complexity of higher resolution images may require additional adjustments to the model's hyperparameters, such as layer depth or latent space size. This is why we increased the latent space to 200, the number of epochs to 50 and the batch to 45, but the images generated were still black (given that this second attempt was a failure and generated exactly the same result as the previous one, we didn't consider it relevant to include it in the outputs). We therefore wondered whether the problem might have arisen during the normalisation or pre-processing of the images: this cannot be ruled out, even if the example of the image normalised during this stage was successful. To be more precise, when the model was trained, the loss indicated "nan" (not a number) for all epochs. This problem can occur for a number of reasons, including when pixel values are incorrectly normalised or when mathematical operations such as division by zero occur. In this case, it would perhaps be a good idea to check the normalisation process carefully to ensure that it is applied correctly and that there are no NaN or infinite values introduced into the data. Despite these considerations, we were unable to identify the source of the problem.



```
xx = (xx * 255).astype(np.uint8)
```

Figure 4: epochs=30, batch-size=25, latentsdim = 100, img = cv2.resize(img, (128, 128))

- Finally, we explored new territory when evaluating the model with a new letter, *alpha*. This allows us to see if we encounter the same problems as with epsilon. The choice of alpha is not insignificant: in fact, this letter has more distinguished shapes on the *papyri*. We therefore wanted to see if our model captured the different representations of a letter during learning and generation. This was not really possible with *epsilon*, since this Greek letter has very few variations in shape when written on *papyri*. It is interesting to note that our model manages to generate various forms of *alpha* consistently. However, it is clear that there are still areas for improvement, notably the sharpness of the contours and the fidelity of the colours in the images generated, as with *epsilon*. These observations suggest that the model effectively captures the structural variations of the alpha letter but could benefit from adjustments to refine detail and improve the overall visual quality of the results.

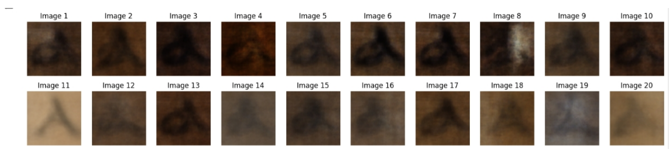


Figure 5: epochs=30, batch-size=35, latentsdim = 100

0.4 Problems encountered

During the development of the Variational Autoencoder (VAE) model for generating images of the letter E, several challenges were encountered. The main difficulty lay in the visual quality of the images generated: despite attempts to adjust hyperparameters such as latent space, number of epochs and batch size, the characters generated often remained unclear and lacked precise detail. In addition, colour management problems were observed, with an unexpected tendency towards blue tones in the images generated. Increasing the image resolution to

128x128 pixels also posed challenges, leading to results where the generated images were all black, suggesting difficulties in managing the increased complexity of high-resolution data.

0.5 Advantages of VAE

The Variational Autoencoder (VAE) has several significant advantages in our project. Firstly, it allows us to efficiently generate new images of the letter E by exploring a latent space where key aspects of the image are represented in an abstract way. This ability to represent and manipulate key features of the image facilitates the generation of diverse and recognisable variations of the letter epsilon from our initial dataset. In addition, the VAE's encoder-decoder architecture enables faithful reconstruction of the original image from latent data, providing flexibility and creativity in image generation. Finally, the efficiency of model training, which does not add significant extra time, makes VAE particularly suitable for rapid iterations aimed at improving the visual quality of the results generated. To give just one example, training a model of 100 epochs took just 2m24s on Google Collab with the T4 GPU as the execution mode.

0.6 Comparison with the GANs method

As a reminder, Generative Adversarial Networks (GANs) aim primarily to generate realistic images by training a generator to produce data similar to that in the real dataset, whereas the Variational Autoencoder (VAE) focuses on the structured representation of data in a latent space.

The Variational Autoencoder (VAE) has several distinct advantages over Generative Adversarial Networks (GANs) in the context of our project. Unlike GANs, which can sometimes produce visually impressive results but can be difficult to stabilise and train, VAE offers a more stable and predictable approach to image generation. VAE focuses on the structured representation of data in a latent space, allowing more controlled exploration and better interpretability of the results generated. In addition, the VAE training process is often faster and more straightforward, which is advantageous when rapid iterations are required to adjust and improve the quality of the images generated. In comparison, our experience with GANs for character generation has not produced satisfactory results, particularly in terms of the clarity and visual fidelity of the characters generated. VAE is therefore a robust and effective alternative for generating images of letters, offering both stable quality and ease of implementation in our image recognition and generation project.

0.7 Bibliography

PyImageSearch. A Deep Dive into Variational Autoencoders with PyTorch . Consulté le 2 juin 2024.
`https://pyimagesearch.com/2023/10/02/a-deep-dive-into-variational-autoencoders-`

Adžemović, Momir. Robotmurlock/VariationalAutoEncoder . Jupyter Notebook, 19 janvier 2024.
`https://github.com/Robotmurlock/VariationalAutoEncoder`.

Auto-encodeur variationnel convolutif — TensorFlow Core . Consulté le 14 juin 2024.
`https://www.tensorflow.org/tutorials/generative/cvae?hl=fr`.

Rocca, Joseph. Understanding Variational Autoencoders (VAEs) . Medium, 21 mars 2021.
`https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f705`