Image Retrieval with Autoencoder

Introduction: In this project, the image retrieval problem was addressed from an unsupervised perspective using Autoencoder. The retrieval system relies on the latent space representation of images obtained through network bottlenecks. Here, the goal is to capture latent space embeddings of images and identify images with the shortest Eudiclean distance from the query image. The rand index and normalized mutual information score were calculated to validate the system. Experimental results demonstrate that extracted features are representable and can be used to retrieve relevant images.

A. Methodology

1. Data:

Dataset: For this project, the Google Landmarks Dataset v2 was used [1]. The trainset contains over four million images of various sizes are used as input for the model.

Preprocessing: For scripting and storing the dataset, Google Colab was utilized.

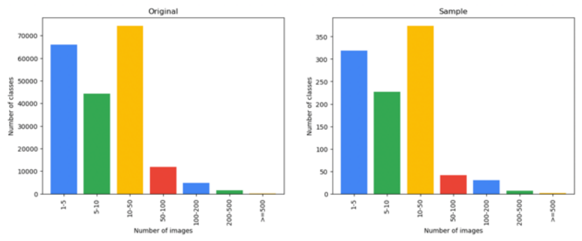


Figure1. Data distribution

First, from the train.csv file, images whose landmark_id fall within the 1000- 2000 range were sampled to save time and computing power. It is visible that the sampled dataset maintains the original distribution. One percent of the resulting dataset was used as the testset. It was ensured that there is at least 1 image for the test set for each class. The remaining 99% of images were then divided into 80/20 training/validation. Images belongs to these datasets are then downloaded into 3 separated folders. While downloading, broken links were removed, and images are resized to 96x96 for computational efficiency.

The downloaded images are then turned into data frames. Incomplete files are removed during this process. The resulting train, validation and test sets have 6186, 1527, and 65 images respectively. For detailed processing, see the DataPreprocessing.ipynb notebook.

2. The retrieval system:

For this project, Pytorch was the main framework in use. The images are put into dataframes and transformed before being put through the neural network. From the autoencoder, the latent features of images was extracted which were then stored for calculation of Euclidean distances. The code for this part can be found in the AutoEncoderRetrieval.ipynb notebook.

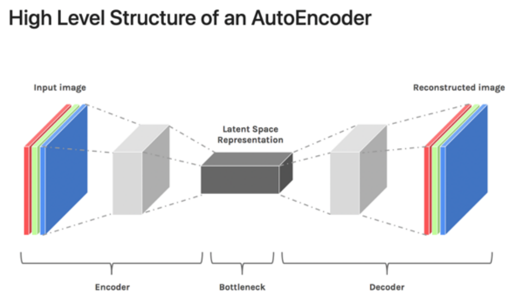

High Level Structure of an AutoEncoder
Input image
Reconstructed image
Latent Space Representation
Encoder
Bottleneck
Decoder

Figure 2. AutoEncoder

Autoencoder: An autoencoder comprises two parts, namely encoder and decoder. The encoder compresses the input image into embeddings wherease the bottleneck of an autoencoder gives rise to the latent space [2]. In the autoencoder, blocks of Conv−Batch Norm−Relu are used. In the encoder part, these blocks are followed by MaxPooling layers. In the decoder, these blocks are preceded by DeConv layers. Latent space embeddings are generated by the last Conv layer in the encoder.

The autoencoder is used on the training and validation data. The Adam optimizer with default parameters wer used. The loss is the mean squared error between the input image and the reconstructed image. The loss in this model is self-supervised as no label information is used.

During the training process, the model with the best validation are stored in pt format for future use.

Euclidean distance: The Euclidean distance between the query image and other images are then calculated based on their latent space embeddings. The less the distance, the more similar images are.

3. Metrics: To measure the performance of the model, the embeddings extracted from training data a clustered together using k-means algorithm. Based on clustering, the MNI and RI metrics were calculated.

Rand Index (RI):

$$\mathrm{RI} = \frac{TP+TN}{TP+FP+FN+TN}$$

where TP, TN, FP, and FN denote true positive, true negative, false positive, and false-negative prediction of cluster labels with respect to the ground truth class labels.

The Rand Index is a measure of the similarity between two data clusterings. RI considering all pairs of samples and counting pairs that are assigned in the same or different clusters in the predicted and actual clusterings. This measure should be high as possible else we can assume that the datapoints are randomly assigned in the clusters.

Normalized Mutual Information (NMI):
NMI is a measure based on mutual information between the true data partition and the obtained clusters.It is a normalization of the Mutual Information (MI) score to scale the results between 0 (no mutual information) and 1 (perfect correlation). Higher values suggest better data clusters [3].

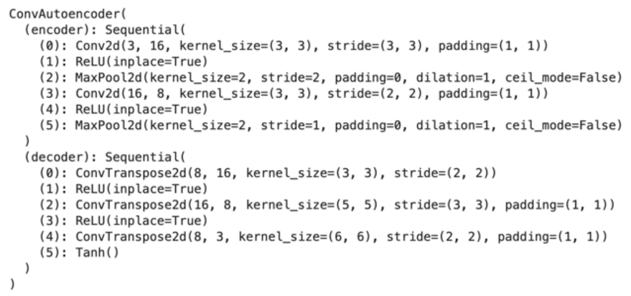
B. Results

Various experiments using three different Autoencoders architectures were conducted. The number of epochs used for these experiments is 20.

Metrics	**Model 1**	**Model 2**	**Model 3**
Best validation loss	0.054003	0.040074	0.024561
NMI	0.139366	0.182436	0.189234
RI	0.925350	0.926809	0.927001

Table 1: Metrics

Model 1: Simple model without batch norm (M1)
This model yields blurry reconstructed images which are not be displayed for conciseness. It also perform poorly in terms of metrics.



```
ConvAutoencoder(
  (encoder): Sequential(
    (0): Conv2d(3, 16, kernel_size=(3, 3), stride=(3, 3), padding=(1, 1))
    (1): ReLU(inplace=True)
    (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (3): Conv2d(16, 8, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
    (4): ReLU(inplace=True)
    (5): MaxPool2d(kernel_size=2, stride=1, padding=0, dilation=1, ceil_mode=False)
  )
  (decoder): Sequential(
    (0): ConvTranspose2d(8, 16, kernel_size=(3, 3), stride=(2, 2))
    (1): ReLU(inplace=True)
    (2): ConvTranspose2d(16, 8, kernel_size=(5, 5), stride=(3, 3), padding=(1, 1))
    (3): ReLU(inplace=True)
    (4): ConvTranspose2d(8, 3, kernel_size=(6, 6), stride=(2, 2), padding=(1, 1))
    (5): Tanh()
  )
)
```

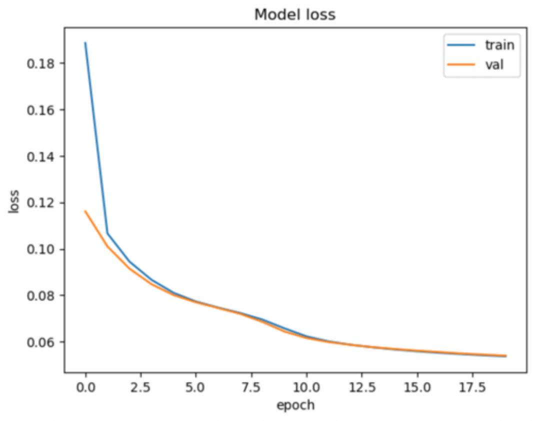
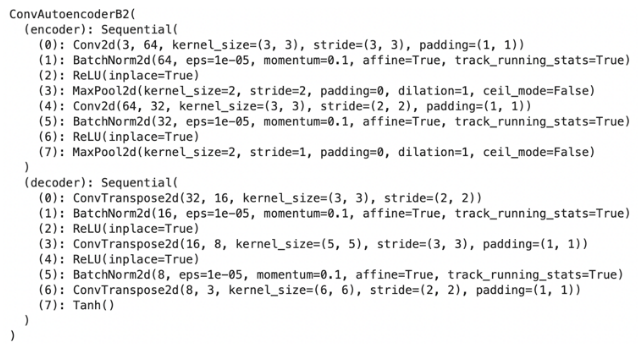


Figure 3. M1 architecture and loss

Model 2: Using batch norm (M2)
This model results in latent space embeddings with only 7 nodes. Thus, the reconstructed images are quite blurry. However, it yields decent NMI and RI, reflecting through good retrieval results just after 20 epochs.



```
ConvAutoencoderB2(
  (encoder): Sequential(
    (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(3, 3), padding=(1, 1))
    (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU(inplace=True)
    (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (4): Conv2d(64, 32, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
    (5): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (6): ReLU(inplace=True)
    (7): MaxPool2d(kernel_size=2, stride=1, padding=0, dilation=1, ceil_mode=False)
  )
  (decoder): Sequential(
    (0): ConvTranspose2d(32, 16, kernel_size=(3, 3), stride=(2, 2))
    (1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU(inplace=True)
    (3): ConvTranspose2d(16, 8, kernel_size=(5, 5), stride=(3, 3), padding=(1, 1))
    (4): ReLU(inplace=True)
    (5): BatchNorm2d(8, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (6): ConvTranspose2d(8, 3, kernel_size=(6, 6), stride=(2, 2), padding=(1, 1))
    (7): Tanh()
  )
)
```

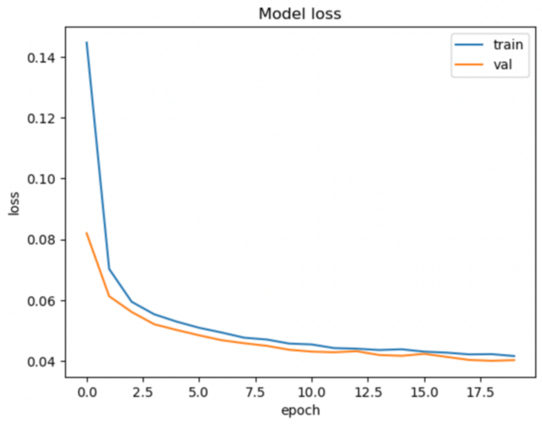


Figure 4. M2 architecture and loss

Reconstruction:

Figure 5. M2 Reconstruction

Retrieval:

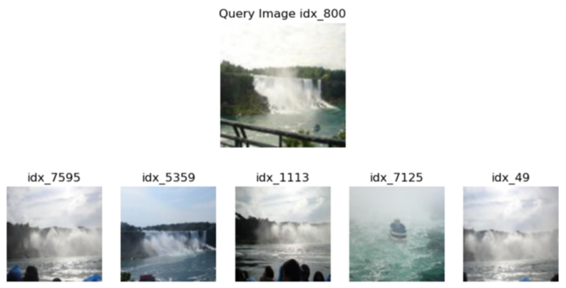


Figure 6a. M2 training and validation retrieval

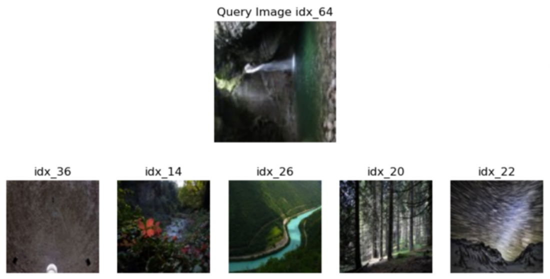
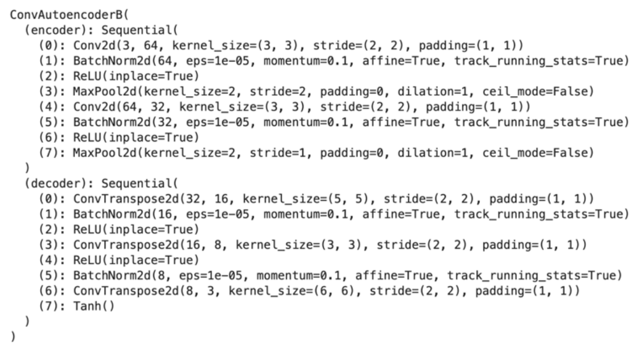


Figure 6b. M2 test set retrievel

Model 3: Using batch norm with increased number of nodes in latent space
This model slightly adjusts the Conv layers of the model above to achieve a higher of number of nodes in the latent space embeddings (increase from 7 to 11). This results in clearer reconstructed images, lower losses, and improved metrics.



```
ConvAutoencoderB(
  (encoder): Sequential(
    (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
    (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU(inplace=True)
    (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (4): Conv2d(64, 32, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
    (5): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (6): ReLU(inplace=True)
    (7): MaxPool2d(kernel_size=2, stride=1, padding=0, dilation=1, ceil_mode=False)
  )
  (decoder): Sequential(
    (0): ConvTranspose2d(32, 16, kernel_size=(5, 5), stride=(2, 2), padding=(1, 1))
    (1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU(inplace=True)
    (3): ConvTranspose2d(16, 8, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
    (4): ReLU(inplace=True)
    (5): BatchNorm2d(8, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (6): ConvTranspose2d(8, 3, kernel_size=(6, 6), stride=(2, 2), padding=(1, 1))
    (7): Tanh()
  )
)
```

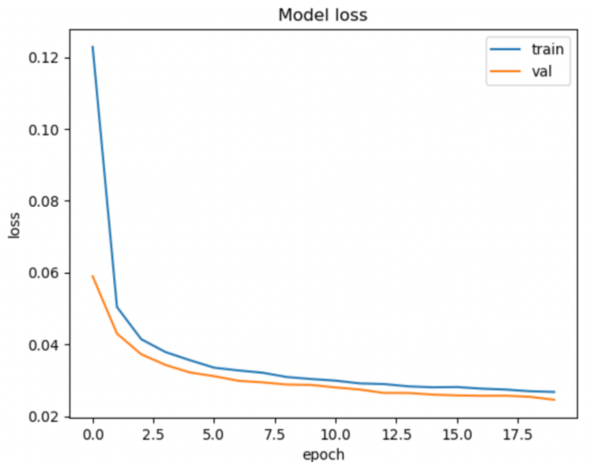
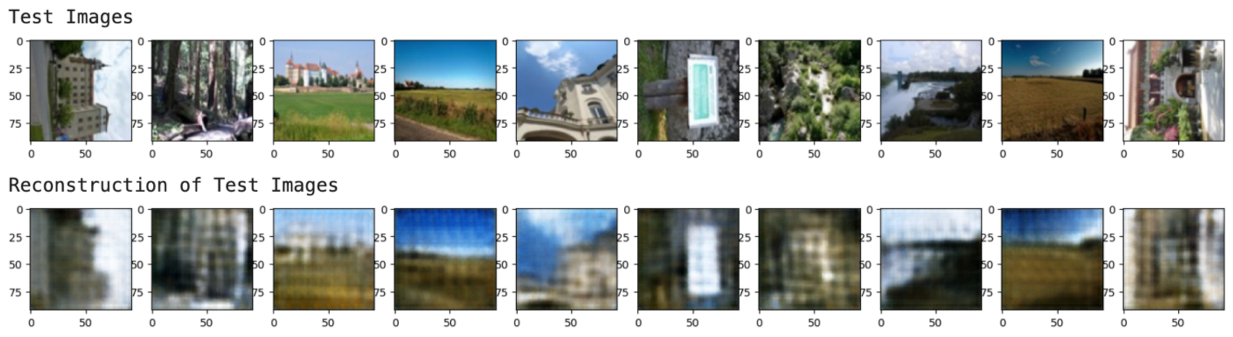
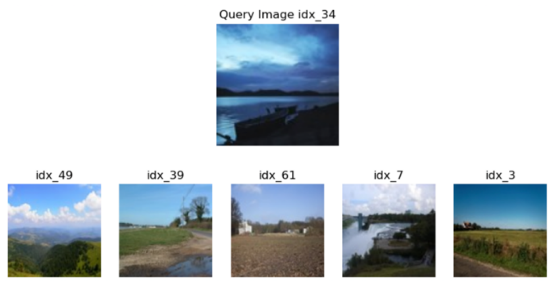


Figure 7. M3 architecture and loss

Reconstruction:





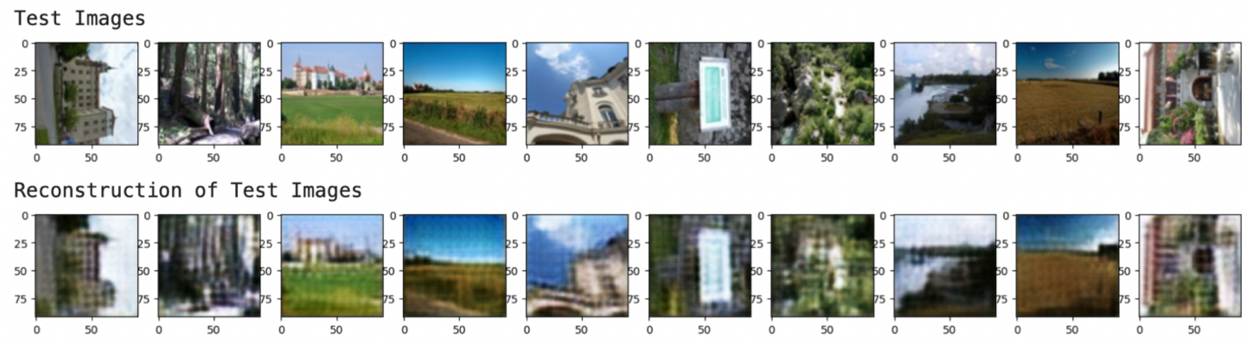


Figure 8. M3 reconstruction

Retrieval:

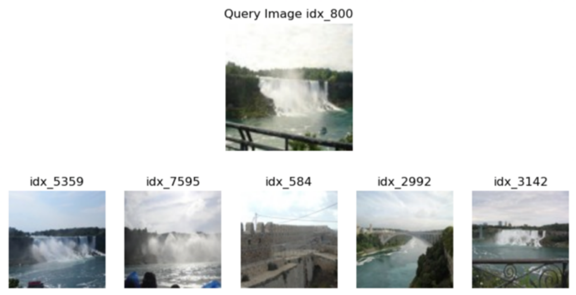


Figure 9a. M3 training and validation retrieval

Figure 9b. M3 test set retrievel

Conclusion:

All of the three models show decreasing training and validation losses, reflecting the ability to learn the image representations. Additionally, the use of batch normalization has been shown to contribute to improved performance. Also, although all models were able to retrieve relevant images, it is evident that models 3 outperforms with the best metrics and clearest reconstructed images. For improvement of the retrieval system, a deeper autoencoder network and more training epochs should be employed.

Citations:

[1] "Google Landmarks Dataset v2 - A Large-Scale Benchmark for Instance-Level Recognition and Retrieval" T. Weyand*, A. Araujo*, B. Cao, J. Sim Proc. CVPR'20

[2] I. A. Siradjuddin, W. A. Wardana and M. K. Sophan, "Feature Extraction using Self-Supervised Convolutional Autoencoder for Content based Image Retrieval," 2019 3rd International Conference on Informatics and Computational Sciences (ICICoS), Semarang, Indonesia, 2019, pp. 1-5, doi: 10.1109/ICICoS48119.2019.8982468.

[3] Gosthipaty, Aritra Roy, and Souradip Chakraborty. “Towards Representation Learning for an Image Retrieval Task.” W&B, August 7, 2020. https://wandb.ai/authors/image-retrieval/reports/Towards-Representation-Learning-for-an-Image-Retrieval-Task--VmlldzoxOTY4MDI.