Compressing Tissue Sample Images Using Machine Learning

Aishwaryaa Udeshi [1] [3], **Dr. Guangyu Wang** [2]

John Foster Dulles High School, Sugar Land, TX[1]

Houston Methodist Research Institute, Houston, TX[2]

Gifted and Talented Mentorship Program, Fort Bend ISD, TX[3]

Introduction

Machine learning (ML) is defined as **the process a machine undergoes to learn patterns in data**, which are then used to make predictions. A supervised ML algorithm is trained on both features (what the algorithm bases its predictions on) and labels (what the correct predictions are for the features). However, an unsupervised ML algorithm is trained without labels, finding patterns among the features by themselves. A self-supervised ML algorithm is a type of unsupervised ML algorithm that generates labels from the unlabeled data itself, and an autoencoder is **a type of self-supervised ML algorithm that encodes data, particularly images, into a smaller space without losing any necessary information**. This reduces the dimensionality of the data by removing unnecessary information, which reduces the complexity of the data, making it less muddled. Autoencoders have numerous applications in the medical imaging world; namely, they are used to assist in diagnoses. For instance, an autoencoder may be used to detect if a tissue is benign or malignant without the distinction even having been made by humans by feeding it images of both and allowing it to separate them into two groups based on what it identifies as the distinguishing factor.

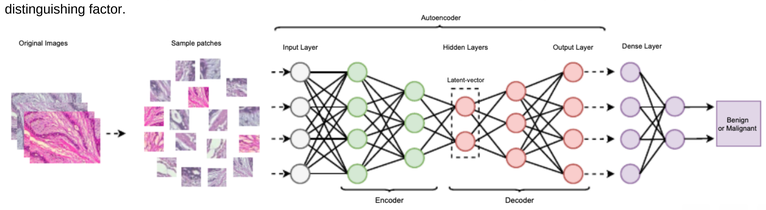

Figure 1. Diagram of a traditional autoencoder in the context of identifying tissue as benign or malignant (Mridha et al., 2021)

A **traditional autoencoder,** also referred to as simply an autoencoder, **consists of an encoder algorithm and a decoder algorithm**. The encoder compresses the pixels into a vector called the latent space, and the decoder attempts to reconstruct the original image. A **Variational Autoencoder (VAE)** also consists of an encoder and a decoder, but the encoder outputs a **latent distribution**, a vector containing the mean and standard deviation of locations for each data point, instead of directly outputting the latent vector. A **sampling layer** converts the latent distribution into the latent vector by sampling data points from a normal distribution and scaling them to the mean and standard deviation in the latent distribution, introducing an element of randomness that allows the model to create more complex, realistic predictions.

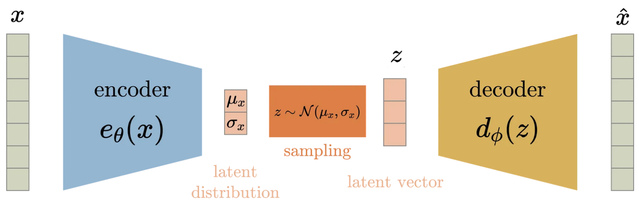

Figure 2. Diagram of a VAE (Anwar, 2021)

Methodology

First, a traditional autoencoder was developed using the MNIST Dataset of Handwritten Digits, which contains 60,000 images of scanned handwritten numbers.

Figure 3. Examples of images from the MNIST dataset

After the images were shuffled randomly and separated into batches of 128 to create smaller groups for the model to process, he autoencoder was trained. Mean Squared Error (MSE) loss was chosen to evaluate the accuracy of the model as it is a standard loss evaluation tool in ML. It performed with the lowest MSE loss with the following parameters (with 10 epochs and the optimizer being Adam):

Encoder

Architecture (Layers)	Activation Function
Flatten	N/A
Dense (x3)	ReLu()

Decoder

Architecture (Layers)	Activation Function
Dense (x3)	Sigmoid()
Reshape	N/A

Methodology (Continued)

Secondly, using the Zenodo “100,000 histological images of human colorectal cancer and healthy tissue” dataset, which contains 7,180 images of human colorectal tissue, both healthy and pathological. The types of tissue included were adipose (ADI), background (BACK), debris (DEB), lymphocytes (LYM), mucus (MUC), smooth muscle (MUS), normal colon mucosa (NORM), cancer-associated stroma (STR), and colorectal adenocarcinoma epithelium (TUM). A VAE was built and trained in Google Colab, an online Python Integrated Development Environment (IDE). First, the images were preprocessed by resizing them to the same shape, which was (64, 64, 4). To merge them into the final training dataset, the images were shuffled randomly and separated into batches of 128.

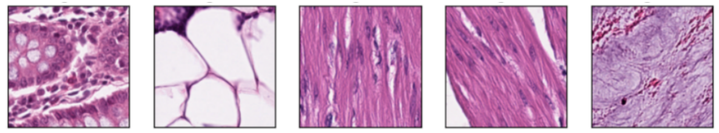

Figure 4. Examples of preprocessed images from the Zenodo dataset

Both the traditional autoencoder and the VAE were built as Keras models using the Tensorflow and NumPy libraries, the former being an open-source ML platform and the latter being a Python package that reduces runtime for numeric computations. In order to allow for the visualization of the latent space, the number of latent dimensions was set to 2. For the VAE, in addition to MSE loss, Kullback-Leibler (KL) Divergence loss was also chosen because of its ability to measure the difference between two probability distributions. The VAE performed with the lowest MSE and KL Divergence losses with the following parameters (with 1 epoch of 57 steps and the optimizer being Adam(0.0001, 0.5)):

Encoder

Architecture (Layers)	Activation Function
Conv2D (x5)	LeakyReLU(0.02)
Flatten	ReLu()
Dense	SeLU()

Decoder

Architecture (Layers)	Activation Function
Dense (x2)	SeLu()
Conv2DTranspose (x4)	LeakyReLU(0.02)
Conv2DTranspose	Sigmoid()

Note: Batch normalization was performed between each layer in both the encoder anf decoder

Results

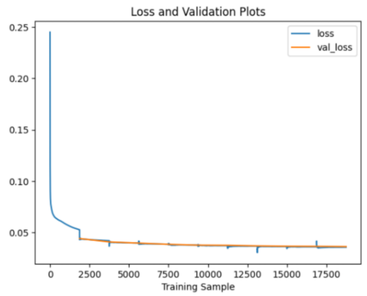

Figure 5. Loss and Validation Plots for the traditional autoencoder

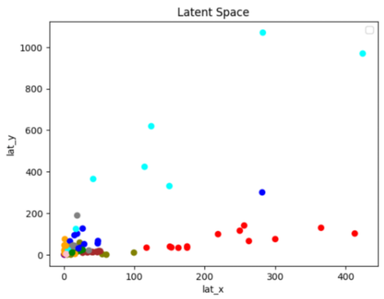

Figure 6. Latent Space for the traditional autoencoder

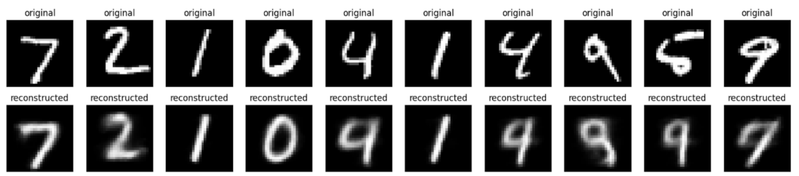

Figure 7. Original vs. Reconstructed Image Comparisons for the traditional autoencoder

Results (Continued)

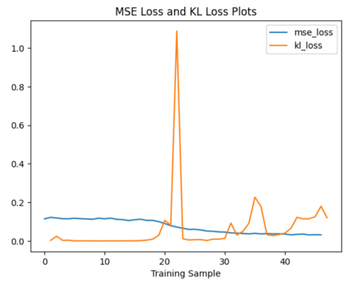

Figure 8. MSE and KL Divergence loss plots for the VAE

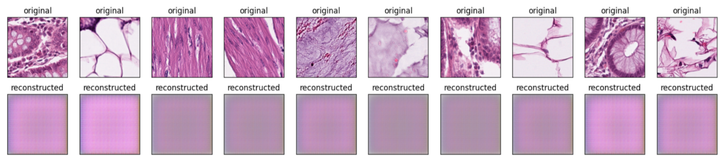

Figure 9. Original vs. Reconstructed Image Comparisons for the VAE

Findings/Conclusion

The traditional autoencoder performed far better than the VAE; specifically the MSE loss of the former was 0.0674 while the latter’s was 0.312. In other words, the traditional autoencoder performed over 4.5 times better than the VAE. However, Figure 6 shows that there are still many improvements to be made. While some numbers clustered together strongly (such as 1, represented by orange, 9, represented by gray, and 2, represented by olive), others were spread out throughout the graph (such as 0, represented by red, 7, represented by cyan, and 4, represented by blue). The model was less able to identify the specific sets of features characterizing the latter category of numbers, in turn making its ability to reconstruct them poorer. This is reflected in Figure 7 as well, where 4 was consistently reconstructed as 9. The autoencoder’s losses decreased throughout training as shown in Figure 5; therefore, the model displays no signs of overtraining.

As for the VAE, Figure 8 shows the MSE loss to have steadily decreased throughout training, but the KL loss to have spiked at times. This means that while the model was improving throughout the process, the latent distribution differed widely from the predefined distribution at some points. Figure 9 shows that the model was able to identify different shades as a feature and account for those in its reconstructed images but failed to pick up edges and contours.

Discussion

In the future, both models would benefit from data augmentation, which involves rotating and scaling the current images to create new ones. This creates more data for the models to train on without needing new images. This would be particularly useful for the VAE as the number of images for each type of tissue is slightly unbalanced as noted by the authors of the dataset. Both models would also benefit from an increase in the number of latent dimensions; for this study, it was set to 2 to allow for the visualization of the model’s groupings. However, realistically, this number does not need to be as small as 2 despite the goal being to reduce the number of dimensions needed to represent an image. With more complex imaging software, the latent space may still be able to be visualized in higher dimensions. Additionally, the VAE would benefit from learning rate scheduling, which adjusts the learning rate as the model trains instead of manually by the researcher after they analyze the results. This was especially an issue with the computational limitations posed by the equipment used in this study, which slowed the speed of training and therefore slowed the process of adjusting the parameters to improve the accuracy. Finally, experimenting with different types of neural networks such as CNNs within the encoders and decoders would likely lead to gains for both algorithms.