AN ANALYSIS OF COMPRESSIVE CONVOLUTIONAL AUTOENCODERS FOR IMAGE ARCHIVING IN MEDICAL INFORMATICS

By

Charles I. Warren

A THESIS

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

In Cybersecurity

MICHIGAN TECHNOLOGICAL UNIVERSITY

2022

This thesis has been approved in partial fulfillment of the requirements for the Degree of MASTER OF SCIENCE in Cybersecurity.

Department of Computer Science

Thesis Advisor:	*Guy Hembroff*
Committee Member:	*Jeff Wall*
Committee Member:	*Xiaoyong Yuan*
Department Chair:	*Jean Mayo*

Table of Contents

List of Figures

Acknowledgements

Thanks to Dr. Guy Hembroff for his never-ending patience.

Thanks to my wife, Jodi, for letting me work full-time and do a Master's.

Thanks to Steven Whitaker for listening to my autoencoder woes and helping debug code.

List of Abbreviations

CNN - Convolutional Neural Networks

MLP – Multilayer Perceptron

SSIM – Structural Similarity Index Metric

MSE – Mean Squared Error

CAE – Convolutional Autoencoders

BCE – Binary Cross Entropy

Abstract

Within a given enterprise network, an array of data types needs to be communicated. These network transmissions consist of images, videos, text, and binaries that have unique requirements of bandwidth and computational overhead to transmit. With respect to medical informatics, these include a multitude of varying subjects, standards, and modalities which are communicated to and from imaging equipment, clinicians, and medical archives. To reduce the required bandwidth to transmit, or provide adequate storage capacity for archival purposes, the data may be compressed in such a way that reduces the size of the image when it is transferred or stored. The original data may be reconstructed either completely or to an acceptable degree of completeness using lossy or lossless compression strategies. The scope of this inquiry is to define ways in which convolutional compressive autoencoders may be used for lossy compression. Multiple approaches will be identified and introduced to define their respective optimal datasets, along with their tuned hyperparameters.

Introduction

In medical informatics, archives are composed of various forms of imagery including MRI, X-Rays, Digital Pathology, and CT artifacts. These various forms of imagery must be maintained for between 5 to 10 years in the United States depending on the state of origin. In order to reduce the required storage of medical information, compression must be used. Forms of compression which can achieve greater than 3:1 are most often lossy algorithms which can have a significant impact on the ability of clinicians to perform an accurate diagnosis.

As misdiagnosis can prevent patients from receiving timely treatment or cause significant detriment to their condition, it is critical that any lossy compression algorithms are used after a decision has already occurred and for archival purposes only. Ideally, the image be maintained in a form where it can be returned to its original state until the physician is confident that they have pursued the correct course of action for that patient.

The problem aimed to be solved by this inquiry is to reduce the necessary bandwidth and storage required to transmit and retain medical imaging for archival purposes.

In consideration of why this is important in the field of medical information, a quote from the Society of Imaging Informatics in Medicine:

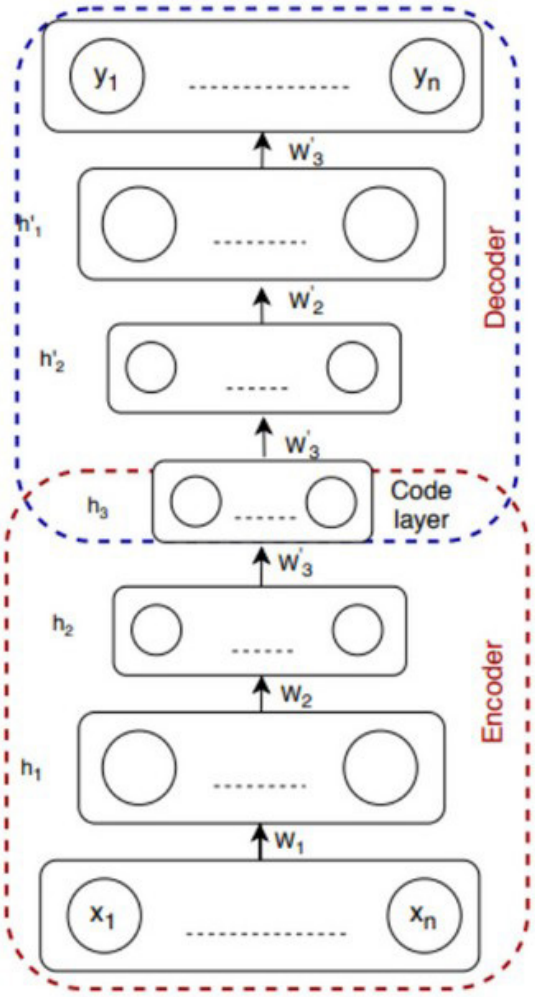
"Medical image archives can grow to contain enormous amounts of data. Radiology, Cardiology, or Pathology image sets can consume several gigabytes of storage. The sum total of image data associated with many patients and over many years can run into the petabyte (1015bytes) range. Obviously, a practical medical image archive must economize on the cost of raw storage." [10]

The solution to this problem needs to be compliant with current medical imaging standards of error, though such standards are most certainly internationally vague [3] and would depend greatly on the application. Any lossy compression algorithm should be evaluated on its effectiveness and its ability to not impact or interfere with clinical decision modeling algorithms in the instance where an archive must be restored to confirm a diagnosis.

Related Works

To gain an understanding of how autoencoders may solve this problem, we should first define what an autoencoder is. An autoencoder is composed of an encoder, which is meant to reduce the dimensionality of the feature map, and a decoder, which is meant to reconstruct the feature map to the best of its abilities, and a layer known as “code” which is where they overlap. [13]

Figure 1-1. Autoencoder Architecture [13]



Neural networks are used in the creation of autoencoders, due to their ability to identify relationships and reduce dimensionality with little modification. In the following experiments, CNN based autoencoders will be evaluated over a variety of imaging modalities used in the field of medicine.

CNNs are similar to MLPs, but they are not fully connected, which allows for reduction of dimensionality. The convolutional layer from which their name is derived, breaks up an image into a feature map of smaller size by relying on the idea that the subject being communicated is determined by the presence of smaller

patterns within it. The size of those patterns is determinedby the convolutional kernel which is a square defined in each layer. With only convolutions, and assuming zero padding, one can slowly reduce the size of an image down to smaller components by trimming the edges. But this takes hundreds of layers, when evaluating a high-resolution subject, such as a medical MRI image. To avoid having as many parameters, stride can be configured to reduce the subject resolution at a much faster rate. Stride is especially useful in down sampling as it allows for more flexibility with the kernel, resulting in skipping over data that is directly neighboring the previous convolution. While an efficient method of down sampling, it does introduce the possibility for significant loss and should be used sparingly.

The analysis done in "Deep Medical Image Reconstruction with Autoencoders using Deep Boltzmann Machine Training" [13], shows that there are promising studies beingdone in this area specific to medical informatics.

Table 1.1. Relationship of modality and anatomy in relation to image size [4]

Modality	Anatomy	Image Dimensions (x, y, z, t)	Bit Depth	Uncompressed File Size
Radiography	Chest	(2000, 2500, -, -)	10–16 bits	10 MB
CT	Abdomen	(512, 512, 500, -)	12–16 bits	250 MB
	Brain	(512, 512, 300, -)	12-16 bits	150 MB
	Heart	(512, 512, 100, 20)	12-16 bits	1 GB
Breast Tomosynthesis	Breast	(2457, 1890, 50, -)	10–16 bits	0.4 GB
MRI	Abdomen	(512, 512, 100, -)	12–16 bits	50 MB
	Brain	(512, 512, 200, -)	12–16 bits	100 MB
	Heart	(256, 256, 20, 25)	12–16 bits	250 MB
Ultrasound	Heart	(512, 512, -, 50)/s	24 bits (color)	38 MB/s
PET	Brain	(256, 256, 50, -)	16 bits	6 MB
	Heart	(128, 128, 40, 16)	16 bits	1 MB
Digital Pathology	Cells	(30,000, 30,000, -, -)	24 bits (color)	2.5 GB

'The use of data compression in medical imaging is regulated by government organizations and there are guidelines provided by professional societies [94,95]. In the US, commercial distribution of medical devices is regulated by the Food and Drug Administration (FDA). The FDA regulates PACS with capabilities defined to include medical image transfer, display, processing, and storage. In 1993, the FDA issued a guidance statement for "suitability of lossy compression for different medical applications such as primary diagnosis, referral and archiving" [96]. In these guidelines, the FDA did not require the manufacturers to restrict indications for use of PACS devices which incorporate lossy compression but stated that the manufacturers may voluntarily restrict recommended use. These guidelines also stated that "video and hard copy images which have been subjected to lossy compression shall be provided with a printed message stating that lossy compression has been applied, and the approximate compression ratio".' [4]

Per the above quote, the use of lossy compression for medical informatics in the United States is regulated by the FDA, who advise that loss should be minimized as much as possible. Given these circumstances, any models that are generated for use in medical informatics will be required to state their loss, compression ratio, and name for any viewer. The FDA has also stated the following about lossy compression for use in mammography:

"Currently the FDA does not permit images regenerated from lossy compressed data to be used in the same manner as the original mammogram. While not allowed for final interpretation, lossy compressed images of previously obtained mammograms may be transferred to the patient or another medical institution to be used for comparison purposes if the interpreting physician deems that acceptable." [5]

This is a fair point and one that should be considered when interpreting the results of this evaluation. Once a physician has made a clinical decision on the subject image, the compressed form should be accompanied by any notes made by the radiologist. The intention is to use these compression algorithms for archival purposes, but in the event that a decision is called into question, the ability to restore an image to an understandable form is crucial. As to what an ideal quality is, the FDA has not provided much guidance on an acceptable level of loss within a compression algorithm. It seems that any algorithm or process used to perform lossy compression needs to be certified and that a clinician should be able to reach the same decision based on the reconstructed image.

For the purposes of this paper, the acceptable loss identified in "Lossy Compression Techniques for EEG Signals" [12] will be used. The acceptable loss in most medical applications is a loss of 10 percent according to the referenced study.

Understanding the acceptable boundaries of our loss, a method of evaluating the loss for the autoencoders needed to be established. Finding optimal loss functions for CAEs is still an active area of research so it was necessary to include additional experimentation on what loss functions presented an ideal image. The final model designed was influenced by the findings present in "Deep Convolutional AutoEncoder-based Lossy Image Compression." The authors identified that MS-SSIM and MSE along with findings in regard to using PReLU as an activation function [2].

The goal will be to achieve a similar PSNR to that of study [2] and a SSIM loss of less than 10% as suggested by study [12].

Methodology

This study investigates the use of autoencoding on medical images using CNN based compressive autoencoder. The results will include the steps taken and the tuned hyperparameters. The evaluation of the model's effectiveness on medical images will be conducted on images of varying complexity. This complexity includes the number of features required for accurate clinical decision making, and differing size, and depth requirement based on the respective modality.

The first stage of the experiment is defined as Data Collection, in which CT [6], MRI [7], and X-RAY [8] images of varying anatomy will be sourced.

In the second stage, the data will be processed into grayscale NumPy arrays and split into test and training datasets.

In the third stage, a series of CNN-based autoencoders will be trained and validated using shuffling/folding of the training data.

During training phase, the models will be trained until convergence is reached or it is proven that another iteration performs better at that point. At least 25 epochs will be used to assess the models, and up to 100 epochs with early termination of a patience of 10 epochs.

In the fourth stage, test images will be encoded and decoded. During this process, the compression ratio will be assessed by looking at the model summary to see the dimensionality at its smallest point, and the degree of loss will be measured using a combination of MS-SSIM, SSIM, MSE, and BCE to compare the original image to the reconstructed one.

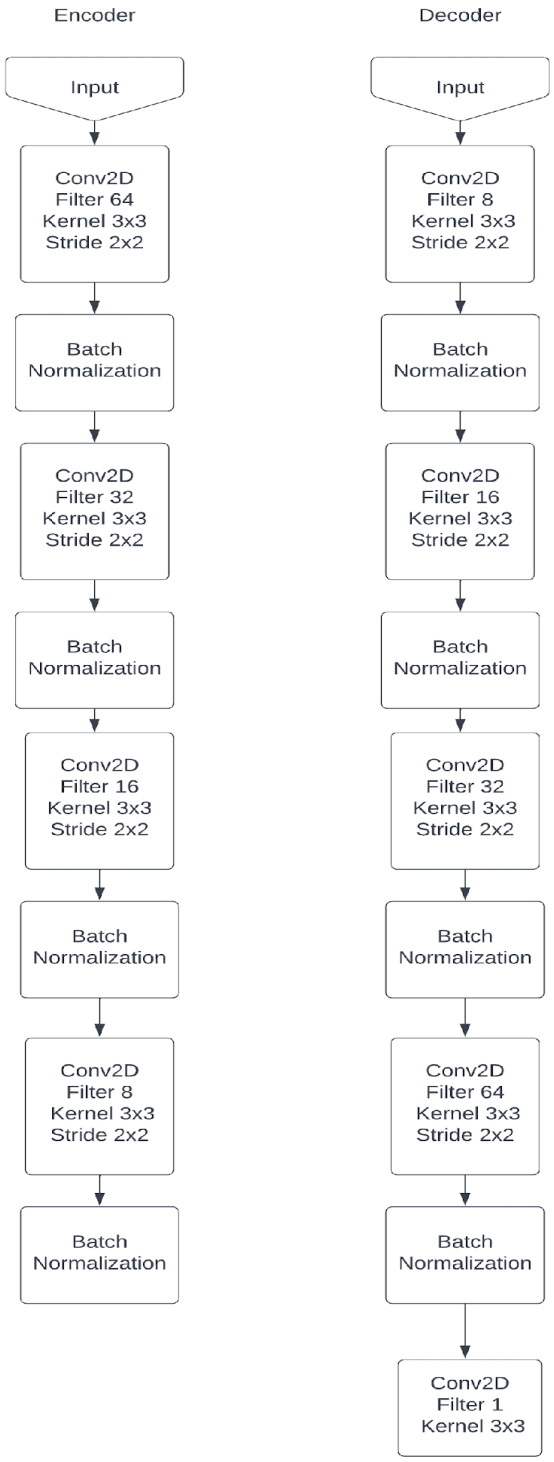
The last step of the experiment will be to use the best model and test its impact on clinical decision making by using an established classification network and providing it with lossy reconstructed test data.

Experiment

Model Selection

Model 1

Figure 2-1. Autoencoder Architecture. See Appendix 1 for more detailed code.



The experiment started by using the first CNN autoencoder with the MRI and CT Brain Scandataset [7]. To identify the base model, the depth of the model was varied from 1 up to 4 layers deep in the encoder and decoder. Modification of the Conv2D and Conv2D transpose stride, kernel, and filter size were made over 25 epochs of testing each in an attempt to reduce binary cross-entropy loss. With the CT training dataset, a value of .1826 BCE loss was achieved with the structure shown in Figure 3-1,

The activation function was also switched between ReLU and tanh, but without significant change being made.

CT Brain Scans:

- Input Dimension: 512x512x1
- Minimum Dimension: 32x32x8
- Compression Ratio: 32:1
- Val Loss: BCE .1826

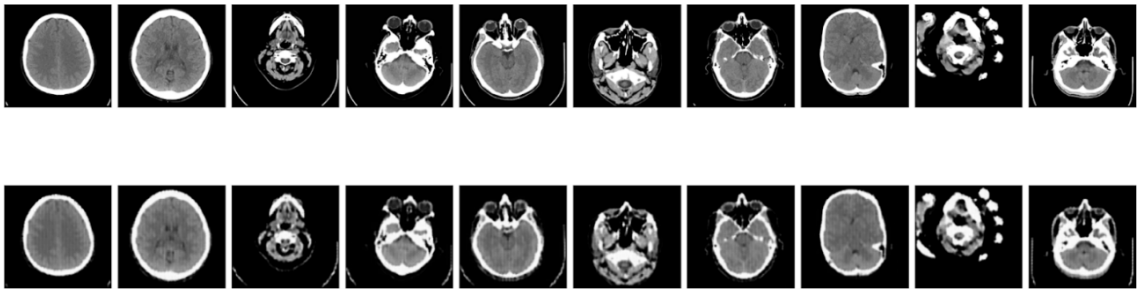


Figure 3-1. CT Brain Scans [7]

MRI Brain Scans:

- Input Dimension: 256x256x1
- Minimum Dimension: 16x16x8
- Compression Ratio: 32:1
- Val Loss: BCE .3587

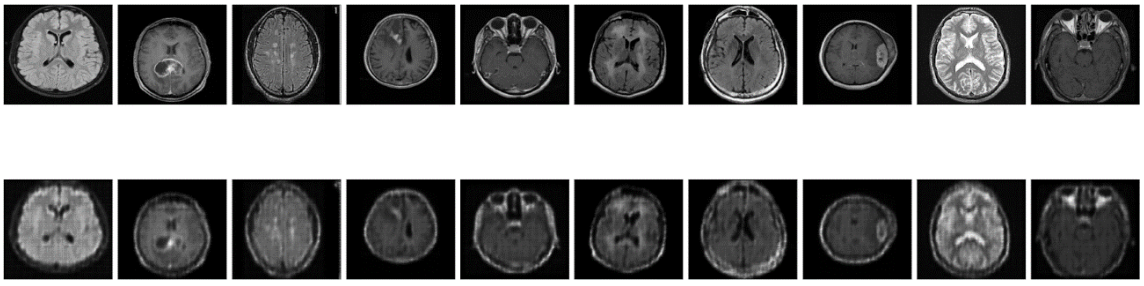


Figure 3-1. MRI Brain Scans [7]

The loss is very apparent in the MRI images, and while the 32:1 compression ratio is very respectable, the loss seems to exceed the usefulness of that compression. The CT images remain visibly different, but the structures are at least recognizable,

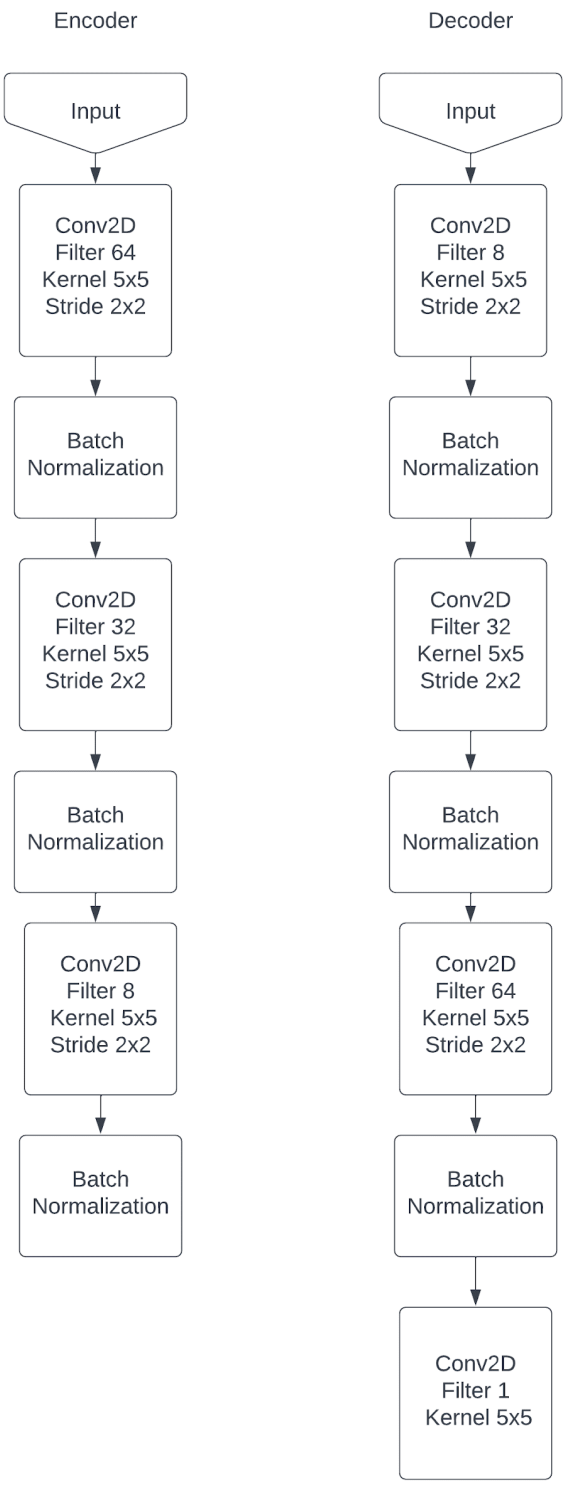
further experimentation will be conducted later in the paper to see if enough information is being preserved.

A medley of loss functions will need to be used keep the model from completely distorting the images, as BSE by itself was not a useful metric. The following loss function was implemented, 0.5 * BSE + 0.25 *MSE + 0.25 * (1-SSIM) based on a considerable amount of trial and error coupled with information from Nvidia's study on SSIM [14], and the previously mentioned studies [13][2].

For the next model the compression was reduced to an 8:1 ratio and the kernel size increased to reduce the total number of features.

Model 2

Figure 4-1. Autoencoder Architecture. See Appendix 2 for more detailed code.



The following are the results of the second architecture over 100 epochs.

CT Brain Scans:

- Input Dimension: 512x512x1
- Minimum Dimension: 64x64x8
- Compression Ratio: 8:1
- Loss:
    - val_loss: 0.0916
    - val_binary_crossentropy: 0.183S
    - val_mse: 0.0016
    - val_ssim_loss: 0.0213

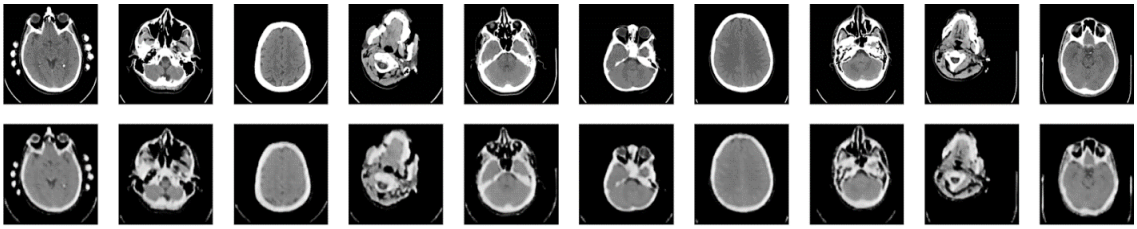


Figure 5-1. CT Brain Scans [7]

MRI Brain Scans:

- Input Dimension: 256x256x1
- Minimum Dimension: 32x32x8
- Compression Ratio: 8:1
- Loss:
    - val_loss: 0.1751
    - val_binary_crossentropy: 0.3502
    - val_mse: 0.0029
    - val_ssim_loss: 0.1021

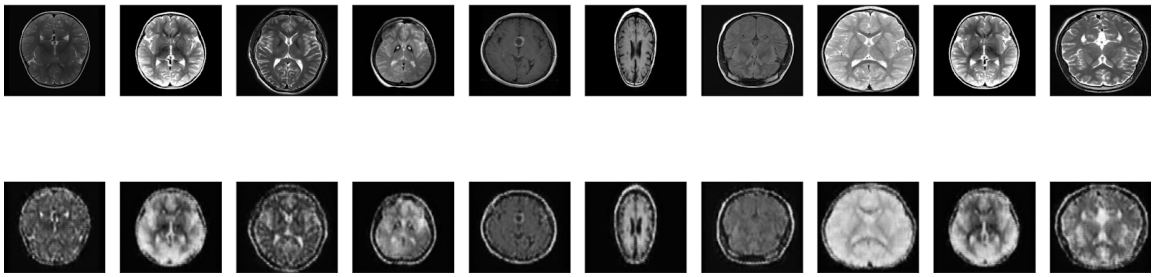


Figure 6-1. MRI Brain Scans [7]

The results of Model 2 are improvements over the Model 1, but at the cost of a reduced compression ratio. In modifying the weights of the new loss function, it was found that adding SSIM increased the prominence of defining borders to patterns within the subject. The contrast was not being preserved with SSIM alone, which is why it was still necessary to create a more complex loss function medley that utilized MSE.

Model 3

Model 3 incorporates the dimensionality reduction of Model 1, the complex loss function of Model 2, and uniquely increases the filter size as more layers are introduced to try and reduce any unintentional bottlenecks.

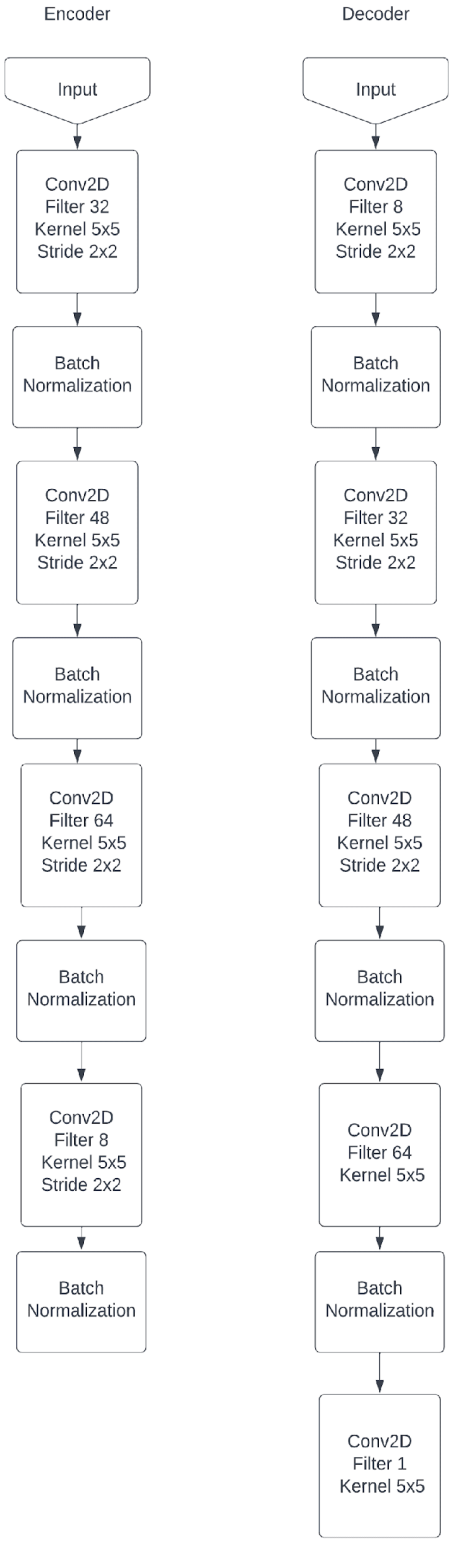


Figure 7-1. Model 3 Autoencoder Architecture. See Appendix 3 for more detailed code.

The following are the results after 100 epochs:

CT Brain Scans:

- Input Dimension: 512x512x1
- Minimum Dimension: 32x32x8
- Compression Ratio: 32:1
- Training Time: 16ms/step
- Loss:
    - val_loss: 0.090
    - val_binary_crossentropy: 0.1802
    - val_mse: 0.0029
    - val_ssim_loss: 0.0365
    - val_psnr: 20.9955 dB

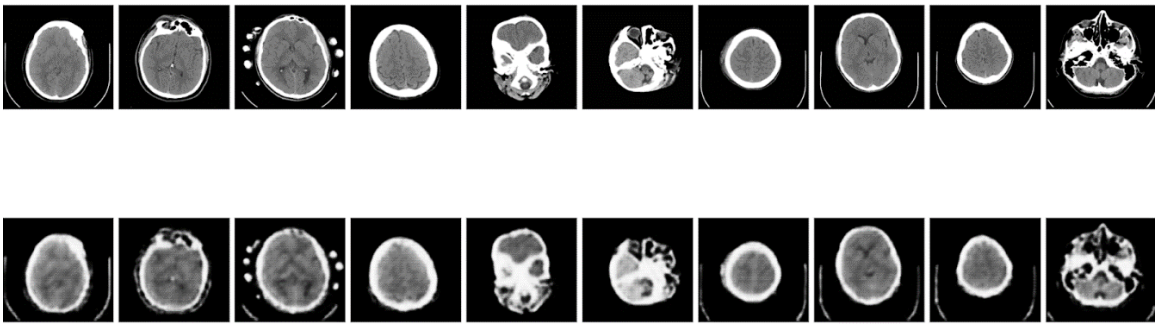


Figure 8-1. CT Brain Scans [7]

MRI Brain Scans:

- Input Dimension: 256x256x1
- Minimum Dimension: 16x16x8
- Compression Ratio 32:1
- Training Time: 7ms/step
- Loss:
    - val_loss: 0.1787
    - val_binary_crossentropy: 0.3574
    - val_mse: 0.0063
    - val_ssim_loss: 0.1674
    - val_psnr: 21.6933 dB

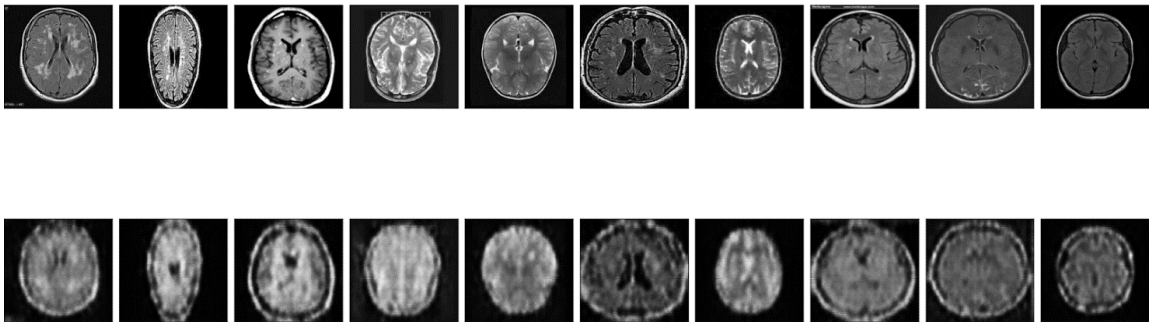


Figure 9-1. MRI Brain Scans [7]

Yet again, the MRI results were underwhelming, but the CT scans provided favorable results.

Some additional datasets were used to evaluate the effectiveness of the model.

COVIDx Chest CT

- Input Dimension: 512x512x1
- Minimum Dimension: 32x32x8
- Compression Ratio: 32:1
- Training Time: 15ms/step
- Loss:
    - val_loss: 0.2468
    - val_binary_crossentropy: 0.4936
    - val_mse: 0.0054
    - val_ssim_loss: 0.2369
    - val_psnr: 23.0740 dB

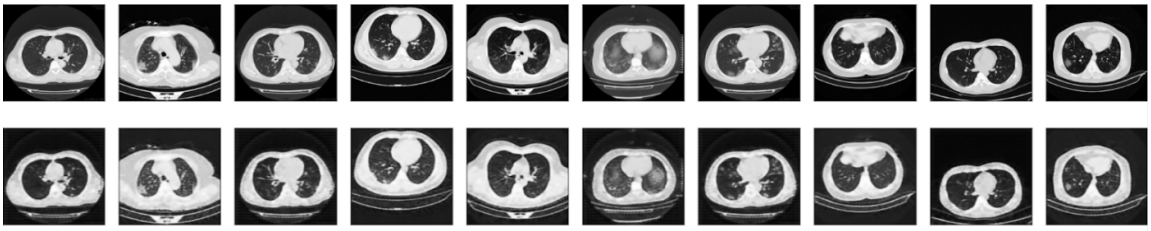


Figure 10-1. COVIDx Chest CT [6]

Chest X-Ray Pneumonia

- Input Dimension: 512x512x1
- Minimum Dimension: 32x32x8
- Compression Ratio: 32:1
- Training Time: 53ms/step
- Loss:
    - val_loss: 0.2769
    - val_binary_crossentropy: 0.5538
    - val_mse: 0.0019
    - val_ssim_loss: 0.0987
    - val_psnr: 26.0472 dB

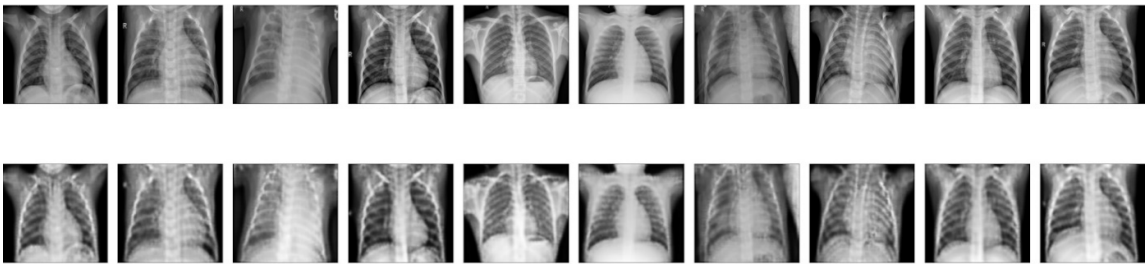


Figure 11-1. Chest X-Ray Pneumonia [8]

Model 4

Model 4 is structurally identical to Model 3, with only changes in its activation function in going from tanh to PReLU and the introduction of MS-SSIM based with MSE in a loss function represented as: 0.5 MS-SSIM * 0.5 MSE. PSNR will be available in addition to MSE and MS-SSIM Loss in order to more accurately assess the model's effectiveness with the removal of BCE.

This model was heavily inspired by the findings in CAE resource [2].

The following are the results after 100 epochs:

CT Brain Scans:

- Input Dimension: 512x512x1
- Minimum Dimension: 32x32x8
- Compression Ratio: 32:1
- Training Time: 22ms/step
- Loss:
    - val_loss: 0.0016
    - val_psnr: 22.5306
    - val_mse: 0.0066

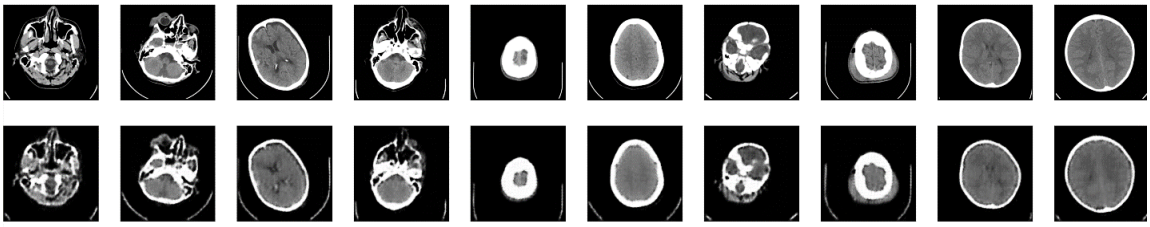


Figure 12-1. CT Brain Scans [7]

MRI Brain Scans:

- Input Dimension: 256x256x1
- Minimum Dimension: 16x16x8
- Compression Ratio 32:1
- Training Time: 15ms/step
- Loss:
    - val_loss: 0.0037
    - val_psnr: 18.9907
    - val_mse: 0.0149
    - val_ms-ssim_loss: 0.2285

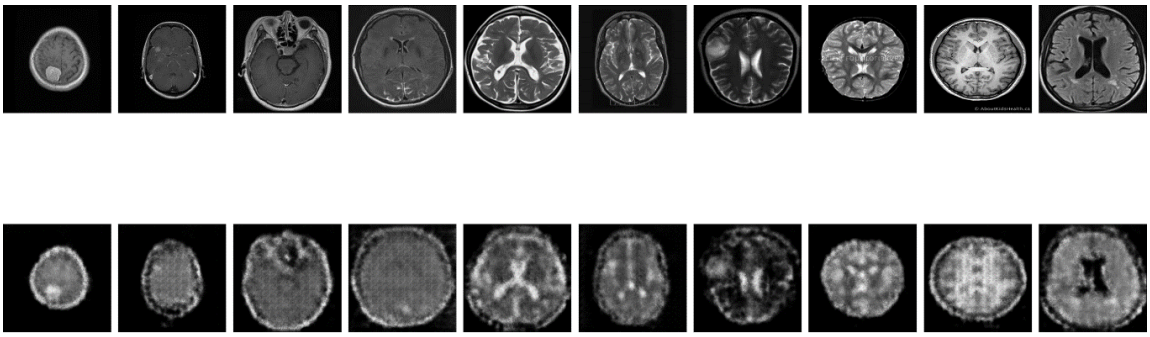


Figure 13-1. MRI Brain Scans [7]

COVIDx Chest CT Scans:

- Input Dimension: 512x512x1
- Minimum Dimension: 32x32x8
- Compression Ratio: 32:1
- Training Time: 31ms/step
- Loss:
    - val_loss: 0.0021
    - val_psnr: 20.9230
    - val_mse: 0.0082
    - val_ssim_loss: 0.1308

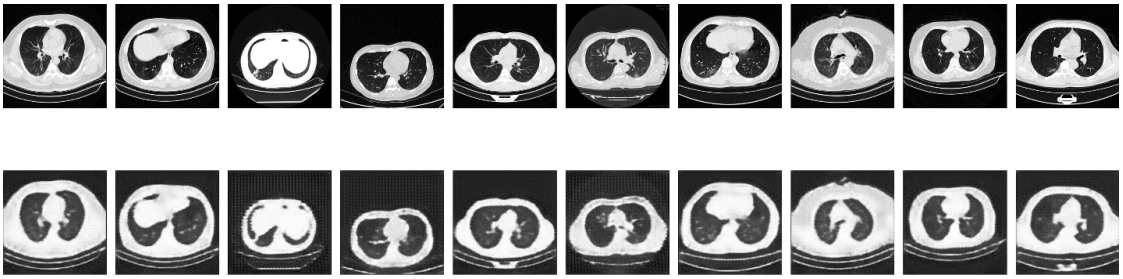


Figure 14-1. COVIDx Chest CT [6]

Chest X-Ray Pneumonia:

- Input Dimension: 512x512x1
- Minimum Dimension: 32x32x8
- Compression Ratio: 32:1
- Training Time: 31ms/step
- Loss:
    - val_loss: 8.0521e-04
    - val_psnr: 25.2433
    - val_mse: 0.0032
    - val_ms-ssim_loss: 0.0897

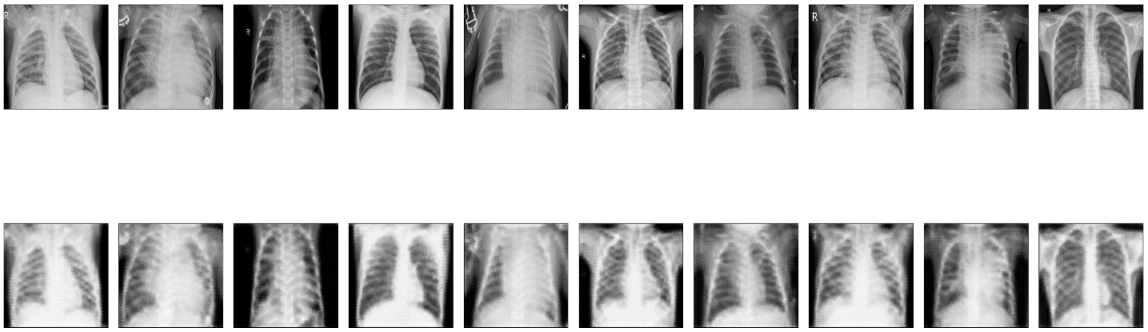


Figure 15-1. Chest X-Ray Pneumonia [8]

Choosing a model:

The expectation was that Model 4 would improve over Model 3, but it became apparent during testing that images were losing their definition significantly. Firstly, by comparing the PSNR across the different datasets:

Table 2.1 Model PSNR comparison

Dataset	Model 1	Model 2	Model 3	Model 4
Brain CT	22.8508	26.6261	20.9955	22.5306
Brain MRI	19.2978	22.3544	21.6933	18.9907
COVID CT	N/A	N/A	23.0740	20.9230
Chest X-Ray	N/A	N/A	26.0472	25.2433

Noting in this figure that PSNR does not tell the whole story and varies significantly in validation runs throughout the experiment. However, using PSNR in conjunction with visual analysis can provide additional information, but is not an indicator of structural information, rather it is an absolute error metric.

A simple visual inspection shows that Model 3 was distorting the images much less than Model 4.

Based on these observations, Model 3 was chosen as the candidate moving forward.

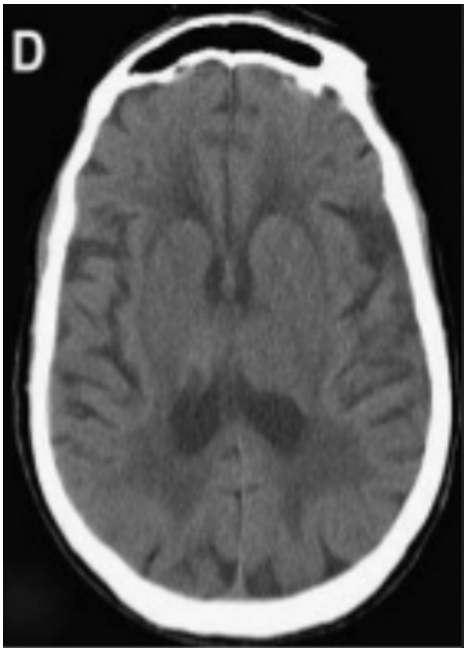


Figure 16-1. Original Figure from the Hemorrhage Dataset [9]

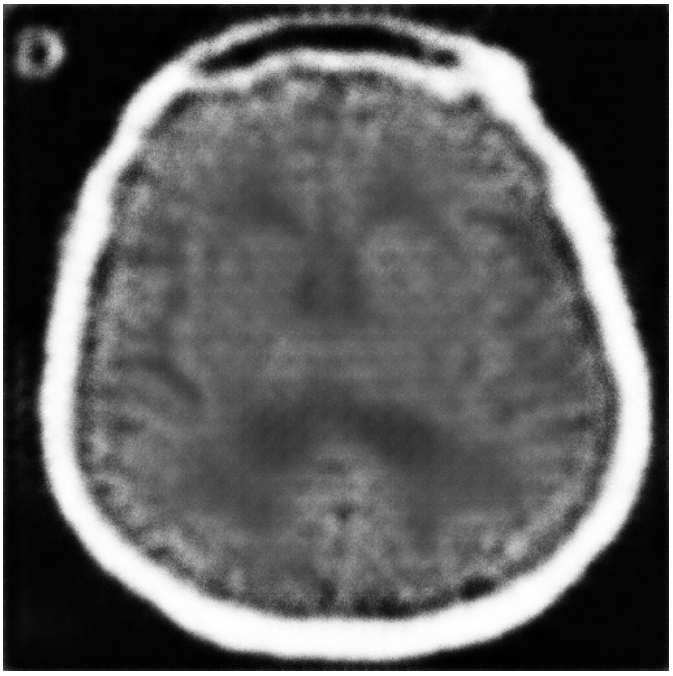


Figure 17-1. Model 3 Trained on Hemorrhage Dataset [9]

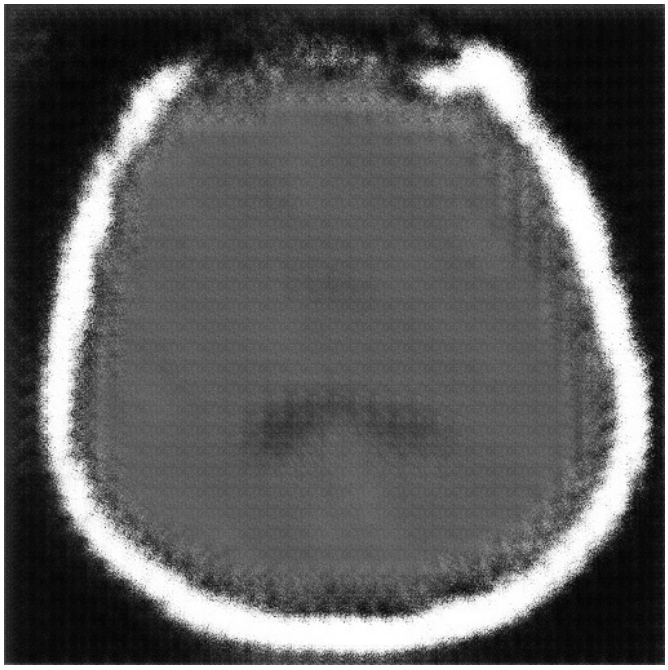


Figure 18-1. Model 4 Trained on Hemorrhage Dataset [9]

Selected Model Validation

Using Model 3 with an open-source notebook [8] intended for the use of classifying pneumonia from chest X-Rays, the following analysis was performed. The notebook was run with the original data first and then with the Model 3 reconstructed test data.

Chest X-Ray Pneumonia

- Input Dimension: 512x512x1
- Minimum Dimension: 32x32x8
- Compression Ratio: 32:1
- Training Time: 16ms/step
- Loss:
    - val_loss: 0.2764
    - val_binary_crossentropy: 0.5527
    - val_mse: 0.0014
    - val_ssim_loss: 0.0536

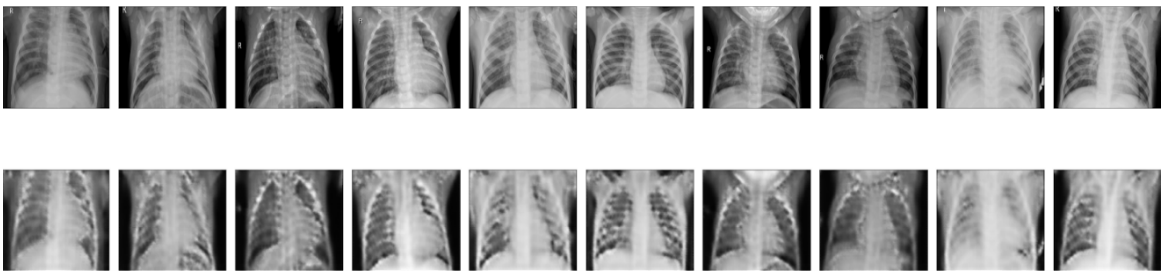


Figure 19-1. Chest X-Ray Pneumonia [8]

Pneumonia Prediction Model Results

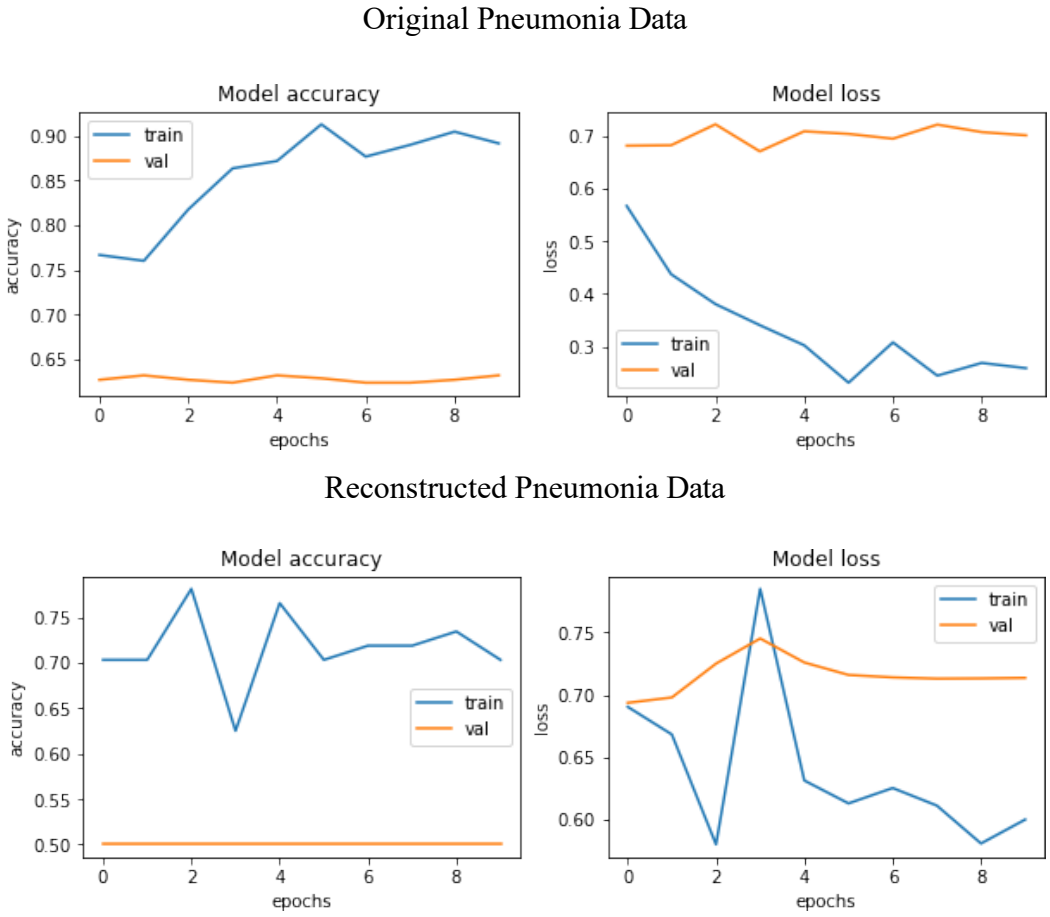


Figure 20-1. Reconstructed Pneumonia Chest X-Ray Model Results

An interpretation of the above graphs shows the training data, which has not been reconstructed, achieve between 75 percent to 90 percent accuracy. The variation between runs is likely due to convergence not occurring until a higher epoch, a change that could be made to the original author's notebook. The important takeaway is the validation accuracy of the second figure which shows no correlation to the training data. This means that reconstructed images do not accurately represent the testing data, which is why these graphs seem to indicate what would be classified as overfitting.

Upon reconstruction, the model is unable to properly discriminate between negative and positive samples. Hoping that this was just a limitation with the dataset, another notebook was used for identifying brain hemorrhages.[9]

Brain CT Hemorrhage

- Input Dimension: 512x512x1
- Minimum Dimension: 32x32x8
- Compression Ratio: 32:1
- Training Time: 16ms/step
- Loss
    - val_loss: 0.2216
    - val_binary_crossentropy: 0.4432
    - val_mse: 0.0055
    - val_ssim_loss: 0.2067

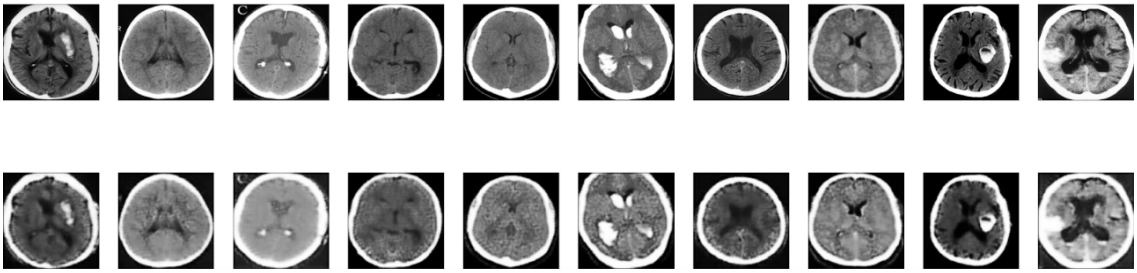


Figure 21-1. Brain CT Hemorrhage [9]

Outcome of Brain Hemorrhage Clinical Decision Model

```
#Now we evaluate on the test set. Remember to make pixels between [0, 1] by dividing by 255.
y_preds = []
for index in range(0,len(test_images)):
    y_preds.append(SIIM_custom_model.predict((test_images[index] // 255).reshape(-1, img_height, img_width, 1), test_labels[index]))

m = tf.keras.metrics.BinaryAccuracy()
m.update_state(test_labels, y_preds)
print(f'Result of original test data: {m.result().numpy()}')

for index in range(0,len(test_images)):
    y_preds.append(SIIM_custom_model.predict((recon_test_images[index] // 255).reshape(-1, img_height, img_width, 1), recon_test_labels[index]))

m = tf.keras.metrics.BinaryAccuracy()
m.update_state(recon_test_labels, y_preds)
print(f'Result of reconstructed test data: {m.result().numpy()}')
✓ 1.3s
Result of original test data: 0.75
Result of reconstructed test data: 0.5483871102333069
```

While the results were not optimal, the fact that there was still a positive correlation in predicting hemorrhages in the reconstructed data insinuates that information required for correct medical diagnosis was maintained to some degree during reconstruction.

Results

An ideal model was identified which could reach a compression ratio of 32:1, up to 96 percent SSIM, and an MSE of less than 0.003. The SSIM of Model 3 is below the 10% loss threshold mentioned in source [12], but the PSNR was significantly lower than that of the models in source [2]. Though most of the datasets upon reconstruction were left with many indiscernible features, through further refinement of the datasets used and the generation of subject, modality, and viewing angle specific models, the loss could be further reduced. The datasets that had more image size variation and were not in a square slide form seemed to struggle more. It would seem that perspective, resolution, and subject should be defining characteristics of each model.

The code and data used to generate these results can be found at the following GitHub link: https://github.com/ciwarren/CCAE-MI

Conclusion

The ability to minimize loss while still providing a compression ratio greater than 16:1 would significantly reduce the burden of cost that is currently assumed for medical archiving purposes. Within the United States, there is regulation by each state requiring various lengths of data retention. In Michigan, the archives must be maintained for at least seven years [15] and as stated by the Society for Imaging Informatics in Medicine these archives can reach petabytes in size when considering how many patients and how often an institution performs imagery procedures. The size of the primary copy of these archives being reduced could lead to greater ability to provide additional redundancy if the cost of storage were to be reduced by a factor of between 16 to 32, increasing the overall availability and security of these archives.

If autoencoders can become a mainstream source of lossy compression, the impacts in medicine could extend further than just for archival purposes. The use of distributing pretrained models as portable encoders and decoders could allow for advancements in areas, such as the Internet of Medical Things (IoMTs). As stated by the authors of the study of lossy compression on EEGs [11], the impact of compression ratios on body area networks can result in significant power savings. To elaborate further, with a reduction of overhead in data transmissions, low power devices may be able to support encryption, where previously unable, due to the significant decrease in payload size.

Another observation made during this analysis, was that the autoencoders also became their own source of anomaly detection. If an autoencoder was trained on a specific set of images, then during reconstruction, a very large loss was present. This is an indication that the training data is not a good indicator of the test data. This could be a crude method of detecting mutations to data while in archive format or the use of the improper model during reconstruction.

With each of the above conclusions, it is important to be mindful that the utility in medicine of lossy compression is only defined by how well images are able to be assessed after being reconstructed.

Future Works

Much more could be done to expand on this experiment, such as using larger training dataset lengths, modifying filter output sizes, breaking up datasets into specific subject views rather than just generic subjects. The downside is a significant increase in computation required to train the model when more trainable parameters are added, and the increase in more models that are needed datasets are broken up.

Additionally, clinicians should be surveyed and presented with varying degrees of loss to an image and their responses should be collected to generalize a threshold to what point a condition, such as a disease, is unrecognizable within the image. This survey would prove very useful in removing some of the ambiguities that exist around lossy compression in medical informatics and may provide the FDA with an ability to provide more achievable benchmarks for new compression alternatives.

Reference List

1. LOSSY IMAGE COMPRESSION WITH COMPRESSIVE AUTOENCODERS, Lucas Theis, Wenzhe Shi, Andrew Cunningham& Ferenc Huszar, Published as a conference paper at ICLR 2017
2. Deep Convolutional AutoEncoder-based Lossy Image Compression, Zhengxue Cheng, Heming Sun, Masaru Takeuchi∗, and Jiro Katto Graduate School of Fundamental Science and Engineering, Waseda University, Tokyo, Japan
3. Guideline for the Use of Image Compression in Diagnostic Imaging, The Royal Australian and New Zealand College of Radiologists, Sydney, Australia, 2020
4. The Current Role of Image Compression Standards in Medical Imaging, Feng Liu, Miguel Hernandez-Cabronero, Victor Sanchez, Michael W. Marcellin and Ali Bilgin, 2017
5. "Can a facility use lossy compression to transmit images to the patient or other medical institutions for final interpretation?" Policy Guidance Help System - Can a facility use lossy compression to transmit images to the patient or other medical institutions for final interpretation? [Online]. Available: https://www.accessdata.fda.gov/cdrh_docs/presentations/pghs/Can_a_facility_use_lossy_compression_to_transmit_images_to_the_patient_or_other_medical_institutions_for_final_interpretation_.htm. [Accessed: 30-Mar-2022]. Chest CT: https://www.kaggle.com/hgunraj/covidxct.
6. H. Gunraj, "COVIDX CT," Kaggle. [Online]. Available: https://www.kaggle.com/datasets/hgunraj/covidxct. [Accessed: 30-Mar-2022].
7. Darren, "CT and MRI brain scans," Kaggle, 11-May-2021 [Online}. Available: https://www.kaggle.com/darren2020/ct-to-mri-cgan. [Accessed: 30-Mar-2022].
8. P. Mooney, "Chest X-ray images (pneumonia)," Kaggle, 24-Mar-2018. [Online]. Available: https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia. [Accessed: 30-Mar-2022].
9. F. Kitamura, "Head CT hemorrhage kernel," Kaggle, 30-Oct-2018. [Online]. Available: https://www.kaggle.com/code/felipekitamura/head-ct-hemorrhage-kernel. [Accessed: 30-Mar-2022].
10. R. Glicksman, "Archiving, Chapter 5: Data Storage Management ," *Society for Imaging Informatics in Medicine*. [Online]. Available: https://siim.org/general/custom.asp?page=archiving_chapter5. [Accessed: 01-Apr-2022].
11. G. Higgins, S. Faul, R. P. McEvoy, B. McGinley, M. Glavin, W. P. Marnane, and E. Jones, "EEG compression using JPEG2000: How much loss is too much?," *2010 Annual International Conference of the IEEE Engineering in Medicine and Biology*, 2010.
12. P. Thi Dao, X. Jun Li, and H. Ngoc Do, "Lossy Compression Techniques for EEG Signals," *Open Repository of Auckland Univeristy* . [Online]. Available: https://openrepository.aut.ac.nz/bitstream/handle/10292/11886/ATC-paper-published-version.pdf;jsessionid=C20974A80B8C4CB19B3D5E5DC90F41EA?sequence=5.

13. S. S and S. Juliet, “Deep Medical Image Reconstruction with autoencoders using Deep Boltzmann machine training,” *EAI Endorsed Transactions on Pervasive Health and Technology*, vol. 6, no. 24, p. 166360, 2020.

14. J. Nilsson and T. Akenine-Möller, “Understanding SSIM,” *ARXIV*, 01-Jul-2020. [Online]. Available: https://research.nvidia.com/publication/2020-07_Understanding-SSIM. [Accessed: 10-MAR-2022].

15. “PUBLIC HEALTH CODE (EXCERPT) Act 368 of 1978,” *Michigan Legislature - Section 333.16213*. [Online]. Available: http://www.legislature.mi.gov/(S(cuho2ysbwlfxyix1gzlpo1hr))/mileg.aspx?page=getobject&objectName=mcl-333-16213. [Accessed: 03-Apr-2022].

Appendix

1 - Model 1

```
input_img = keras.Input(shape=(data_dict['IMG_SIZE'], data_dict['IMG_SIZE'], BATCH_SIZE))
x = keras.layers.Conv2D(64, (3, 3), (2,2), activation='tanh', padding='same') (input_img)
x = keras.layers.BatchNormalization()(x)
x = keras.layers.Conv2D(32, (3, 3), (2,2), activation='tanh', padding='same')(x)
x = keras.layers.BatchNormalization()(x)
x = keras.layers.Conv2D(16, (3, 3), (2,2), activation='tanh', padding='same')(x)
x = keras.layers.BatchNormalization()(x)
x = keras.layers.Conv2D(8, (3, 3), (2,2), activation='tanh', padding='same')(x)
encoded = keras.layers.BatchNormalization()(x)

x = keras.layers.Conv2DTranspose(8, (3, 3), (2,2), activation='tanh', padding='same')(encoded)
x = keras.layers.BatchNormalization()(x)
x = keras.layers.Conv2DTranspose(16, (3, 3), (2,2), activation='tanh', padding='same')(x)
x = keras.layers.BatchNormalization()(x)
x = keras.layers.Conv2DTranspose(32, (3, 3), (2,2), activation='tanh', padding='same')(x)
x = keras.layers.BatchNormalization()(x)
x = keras.layers.Conv2DTranspose(64, (3, 3), (2,2), activation='tanh', padding='same')(x)
x = keras.layers.BatchNormalization()(x)
decoded = keras.layers.Conv2DTranspose(1, (3, 3), activation='sigmoid', padding='same')(x)
```

2 - Model 2

```
input_img = keras.Input(shape=(data_dict['IMG_SIZE'], data_dict['IMG_SIZE'], 1))
x = keras.layers.Conv2D(64, (5, 5), (2,2), activation='tanh', padding='same') (input_img)
x = keras.layers.BatchNormalization()(x)
x = keras.layers.Conv2D(32, (5, 5), (2,2), activation='tanh', padding='same')(x)
x = keras.layers.BatchNormalization()(x)
x = keras.layers.Conv2D(8, (5, 5), (2,2), activation='tanh', padding='same')(x)
encoded = keras.layers.BatchNormalization()(x)

x = keras.layers.Conv2DTranspose(8, (5, 5), (2,2), activation='tanh', padding='same')(encoded)
x = keras.layers.BatchNormalization()(x)
x = keras.layers.Conv2DTranspose(32, (5, 5), (2,2), activation='tanh', padding='same')(x)
x = keras.layers.BatchNormalization()(x)
x = keras.layers.Conv2DTranspose(64, (5, 5), (2,2), activation='tanh', padding='same')(x)
x = keras.layers.BatchNormalization()(x)
decoded = keras.layers.Conv2DTranspose(1, (5, 5), activation='sigmoid', padding='same')(x)
```

3 - Model 3

```
input_img = keras.Input(shape=(data_dict['IMG_SIZE'], data_dict['IMG_SIZE'], 1))
x = keras.layers.Conv2D(32, (5, 5), (2,2), activation='tanh', padding='same') (input_img)
x = keras.layers.BatchNormalization()(x)
x = keras.layers.Conv2D(48, (5, 5), (2,2), activation='tanh', padding='same')(x)
x = keras.layers.BatchNormalization()(x)
x = keras.layers.Conv2D(64, (5, 5), (2,2), activation='tanh', padding='same')(x)
x = keras.layers.BatchNormalization()(x)
x = keras.layers.Conv2D(8, (5, 5), (2,2), activation='tanh', padding='same')(x)
encoded = keras.layers.BatchNormalization()(x)

x = keras.layers.Conv2DTranspose(8, (5, 5), (2,2), activation='tanh', padding='same')(encoded)
x = keras.layers.BatchNormalization()(x)
x = keras.layers.Conv2DTranspose(32, (5, 5), (2,2), activation='tanh', padding='same')(x)
x = keras.layers.BatchNormalization()(x)
x = keras.layers.Conv2DTranspose(48, (5, 5), (2,2), activation='tanh', padding='same')(x)
x = keras.layers.BatchNormalization()(x)
x = keras.layers.Conv2DTranspose(64, (5, 5), (2,2), activation='tanh', padding='same')(x)
x = keras.layers.BatchNormalization()(x)
decoded = keras.layers.Conv2DTranspose(1, (5, 5), activation='sigmoid', padding='same')(x)
```

4 - Model 4

```
input_img = keras.Input(shape=(data_dict['IMG_SIZE'], data_dict['IMG_SIZE'], 1))
x = keras.layers.Conv2D(32, (5, 5), (2,2), activation='PReLU', padding='same') (input_img)
x = keras.layers.BatchNormalization()(x)
x = keras.layers.Conv2D(48, (5, 5), (2,2), activation='PReLU', padding='same')(x)
x = keras.layers.BatchNormalization()(x)
x = keras.layers.Conv2D(64, (5, 5), (2,2), activation='PReLU', padding='same')(x)
x = keras.layers.BatchNormalization()(x)
x = keras.layers.Conv2D(8, (5, 5), (2,2), activation='PReLU', padding='same')(x)
encoded = keras.layers.BatchNormalization()(x)

x = keras.layers.Conv2DTranspose(8, (5, 5), (2,2), activation='PReLU', padding='same')(encoded)
x = keras.layers.BatchNormalization()(x)
x = keras.layers.Conv2DTranspose(32, (5, 5), (2,2), activation='PReLU', padding='same')(x)
x = keras.layers.BatchNormalization()(x)
x = keras.layers.Conv2DTranspose(48, (5, 5), (2,2), activation='PReLU', padding='same')(x)
x = keras.layers.BatchNormalization()(x)
x = keras.layers.Conv2DTranspose(64, (5, 5), (2,2), activation='PReLU', padding='same')(x)
x = keras.layers.BatchNormalization()(x)
decoded = keras.layers.Conv2DTranspose(1, (5, 5), activation='sigmoid', padding='same')(x)
```