DR. DNN: VERIFYING THE RELIABILITY OF ARTIFICIAL INTELLIGENCE IN THE MEDICAL FIELD THROUGH GENERATIVE MODEL'S COUNTER EXAMPLES

By

Heidi B Dye

A capstone project submitted for Graduation with University Honors

May 6th, 2022

University Honors
University of California, Riverside

APPROVED

Dr. Christian Shelton
Computer Science

Dr. Richard Cardullo, Howard H Hays Jr. Chair
University Honors

ABSTRACT

This ongoing research seeks to investigate how Deep Neural Networks (DNN) classify input or make a diagnosis (classification) based on an image of a chest x-ray (input). To accomplish this, we seek to understand the input image itself and gauge whether a DNN is making diagnoses for the correct reasons. To effectively communicate the "reason" for the classification, we adjust the input image to reverse the output diagnosis with as small of a change as possible, thus highlighting the explanation and producing a counterfactual x-ray. However, it is necessary to keep this counterfactual x-ray image as a plausible x-ray to be interpretable by a human and gain the trust necessary to use the tool in practice.

Currently this research focuses on how to construct a flexible parameterization of valid chest x-ray images to be used in such explanations. We are training a Variational Autoencoder (VAE) to identify what many chest x-rays have in common to learn how to generate a fake, but convincing x-ray. This involves using a publicly available dataset of actual images of chest x-rays with eight possible medical diagnoses to train with. The VAE encodes and decodes each x-ray from a latent space. This latent space can then be sampled and searched to generate the desired fake x-ray.

This research explores effective methods for training VAEs in this context, particularly how to balance competing objectives of matching historic images and generalizing well, and how to make efficient use of computational resources to speed up training.

ACKNOWLEDGEMENTS

I would like to thank my mentor Dr. Christian Shelton for his extreme patience with me as I dove headfirst into deep learning. Coming into my Capstone, I had zero experience with deep learning, or even machine learning, but Dr. Shelton helped guide me through this difficult subject. He always took the time to make sure I understood the concepts. I have learned a lot from him these past two years, and I am appreciative for his time.

TABLE OF CONTENTS

Introduction 5

Input 8

Constructing the VAE 11

VAE Implementation Details 12

Initial Testing 20

Optimizations 23

What's Ahead 27

Conclusion 28

References 31

INTRODUCTION

Second opinions are not uncommon when it comes to medical diagnosis. Patients who may have life threatening medical condition such as cancer, would want to verify if they do in fact have such a condition. An incorrect medical diagnosis could be deadly in some cases where the diagnosis is time sensitive. According to a study conducted by the Johns Hopkins University School of Medicine, "…an estimated 12 million Americans suffer a diagnostic error each year – 30% of which result in serious injury or even death." Doctors are humans, and humans are not invincible to error. With the recent outbreak of COVID-19, nations everywhere saw an increase in the demand of medical professionals. There were more patients than there were medical resources and professionals. What if there were a way to make the opinion of a medical professional more readily available? What if this medical diagnosis is proven to be more accurate than a human doctor as well so that patients can be more confident that they were given the correct diagnosis? Both of these problems can be solved by the use of Artificial Intelligence (A.I.) in the medical field.

This research, dubbed DR. DNN for its use of Deep Neural Networks, seeks to provide a human-understandable explanation to the medical diagnoses an A.I can give to patients so that a human can understand how the A.I is behaving. This is so anyone, not just someone who is trained professionally in Deep Learning, can understand the reason why the A.I. gives a certain diagnosis over another. It also seeks to provide credibility to using A.Is in the field of medical diagnosing. Proving why an A.I is giving a certain medical diagnosis is important so doctors and patients alike can trust that it is making the correct or incorrect diagnoses for the right reasons. For instance, while training, the A.I might diagnose a patient to have Leukemia, which is the correct diagnosis for that patient, but it might be looking at the curvature of the spine in the x-ray

to determine this. This is not the correct area of the x-ray to focus on to determine that the patient has Leukemia, but the A.I was deemed already to be accurate since it made the correct diagnosis. In practice, if the A.I kept on focusing on the incorrect areas of x-rays, it could possibly be deadly patients, and discredit the use of A.I to make such diagnoses.

To accomplish this explanation, the research focuses on using a Deep Neural Network (DNN), which is a subarea of A.I, to be able to generate a comparative counterfactual example of chest x-rays that could test the decisive skills of an already trained A.I that gives medical diagnosis based on images of chest x-rays. The counterfactual example should be vastly similar to a real image but differ slightly. In order to obtain the comparative example, a Variational Autoencoder (VAE) is trained and utilized.

For the scope of this research, the VAE is the main objective. A VAE is a necessary component of the explanation to an A.I's decisive skills. The counterfactual example produced from the VAE is a useful tool to give an human understandable explanation in the form of an image like an x-ray. Once the VAE is trained off real images that are medical related, it has the ability to generate other images that look similar the images it was trained with, but it will be entirely made up by the model. This generated example is what is used for the counterfactual example. The differences between the original and the generated counterfactual example are then used to generate the human understandable explanation to why a model chose a certain diagnosis over another, and it can also provide verification of a model's accuracy. The explanation comes from the diagnosis given to both the original and counterfactual example. The difference between the actual images, pixel-wise, can then be used to analyze which parts of the image the model is focusing on to give a medical diagnosis.

It is imperative that the VAE is trained as accurately as possible so the comparative example that is generated is as realistic as possible and can pass as a real x-ray of a human chest. This research can later be expanded upon to include the explanation aspect that was mentioned briefly before where the comparative example and the real version of an x-ray are compared together using an already trained model. This model must be trained to give medical diagnosis to the same type of x-rays that were used to train the VAE with.

Note: Later, comparative and counterfactual are used interchangeably

After the VAE is complete, the visual explanation aspect can be achieved through the use of Deep Contrastive Method (CDeepEx) (Feghahati, et al., 2020). The method proposed is able to answer both the question of why the model chose a certain diagnosis and why the model didn't choose diagnosis A over diagnosis B. CDeepEx is a contrastive method that finds a change in the latent space such that a change in the input, the counterfactual example explains why one classification is not the answer. It was capable of detecting that a model trained on classifying hair color on images of celebrities was actually looking at the color of the eyes. For this research, it can provide the visual explanation using the counterfactual generated example generated from the VAE and the original image.

INPUT

Since the goal of this research is to provide credibility to the use of machine learning in medical diagnosing, it needs a credible dataset. For this, we used The CheXpert dataset available from the Stanford ML Group. The dataset includes training and testing partitions, each with real examples of chest x-rays. Each chest x-ray includes eight possible medical diagnosis that were verified by eight board-certified radiologists, with a majority vote to be correct. The x-rays were diagnosed to have the following: (a) Atelectasis, (b) Cardiomegaly, (c) Consolidation, (d) Edema, and (e) Pleural Effusion. Each x-ray can have multiple diagnoses. In the scope of this research, the classifications of the x-rays is not considered when training. The VAE learns strictly off the input x-rays. The dataset was originally made available as a contest for machine learning experts to train a regular neural network to make medical diagnoses on chest x-rays.

There were two versions of this dataset. One was the normal version, where each image was 2828x2320 pixels (figure 1A). The other was the compressed version where each image was 320320 pixels (figure 1B). For sake of time, the compressed version of the dataset was used. Earlier in the research, the larger version was considered and tested but it provided an extra layer of difficulty that proved to not be necessary. The training set included 224000 examples and the validation dataset has 10000 examples.

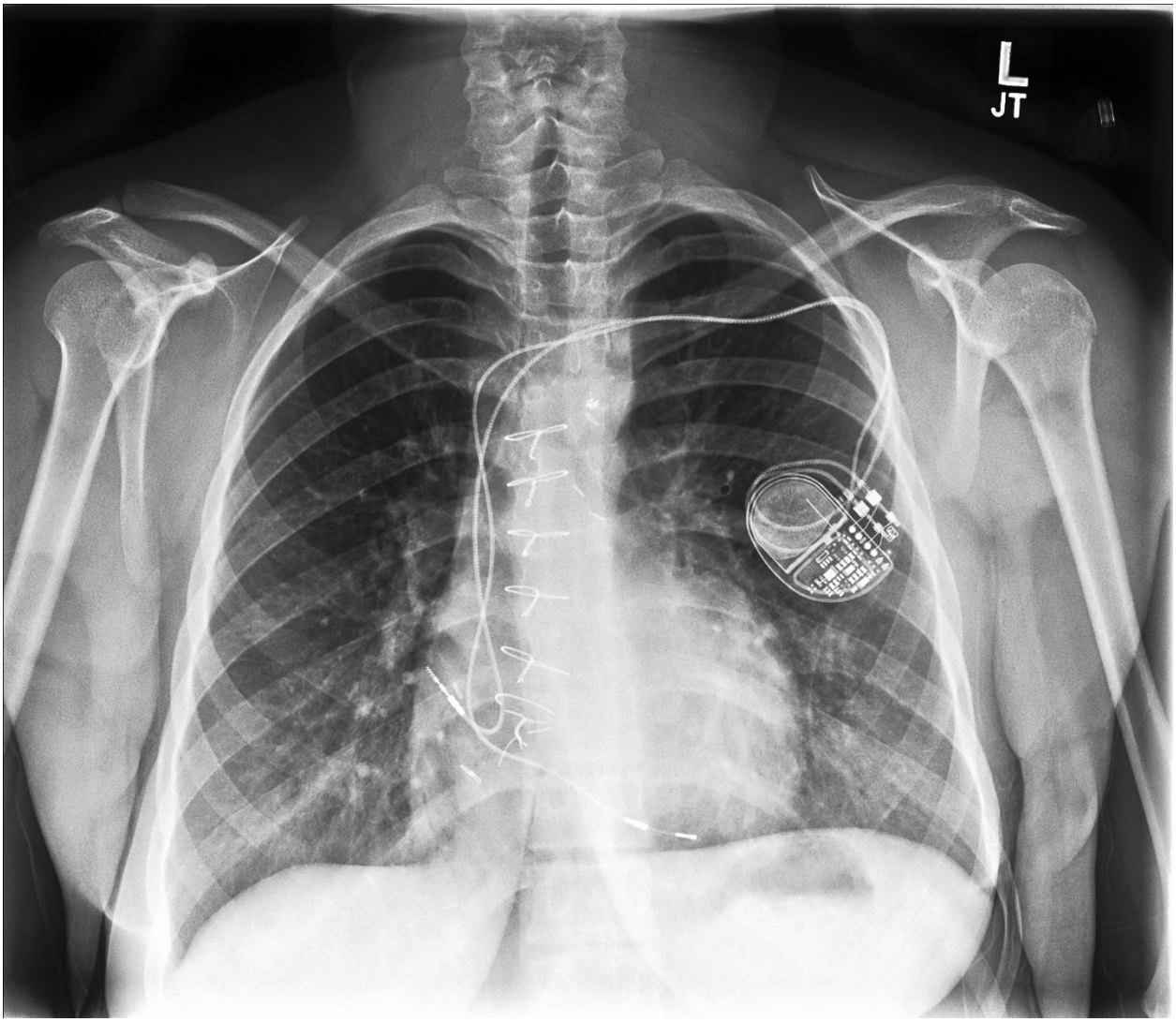


Figure 1A

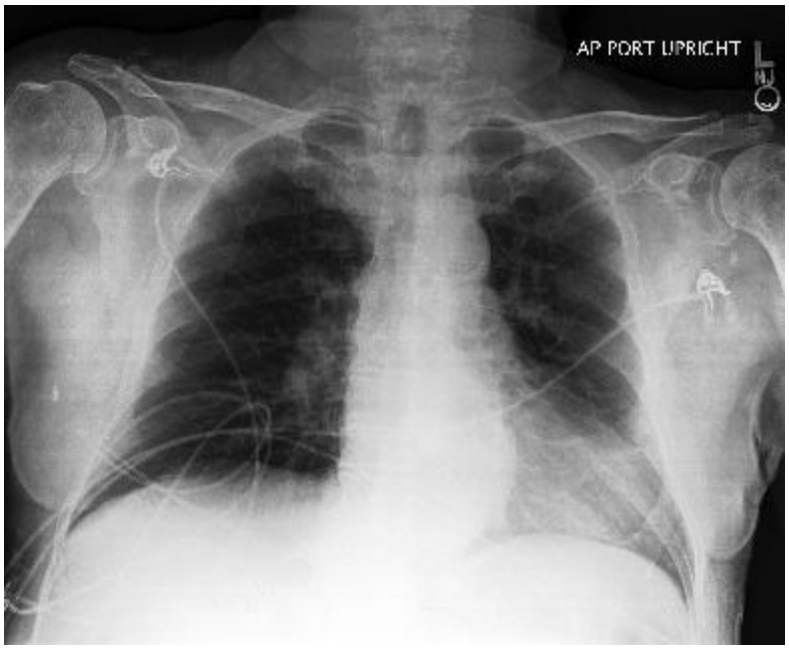


Figure 1B

Note: Both figures are in their original formats, and both represent different x-rays. The smaller version of the x-ray still has its key details evident.

CONSTRUCTING THE VAE

To generate the counterfactual x-ray, we use a type of deep learning method called a Variational Autoencoder (VAE) (Kingma and Welling, 2014). A VAE is generative type of model that is able to learn from input and generate examples that could possibly exist in the original dataset. The VAE takes in a set of images as input and "learns" how to reconstruct these images during the training phase (figure 2A).

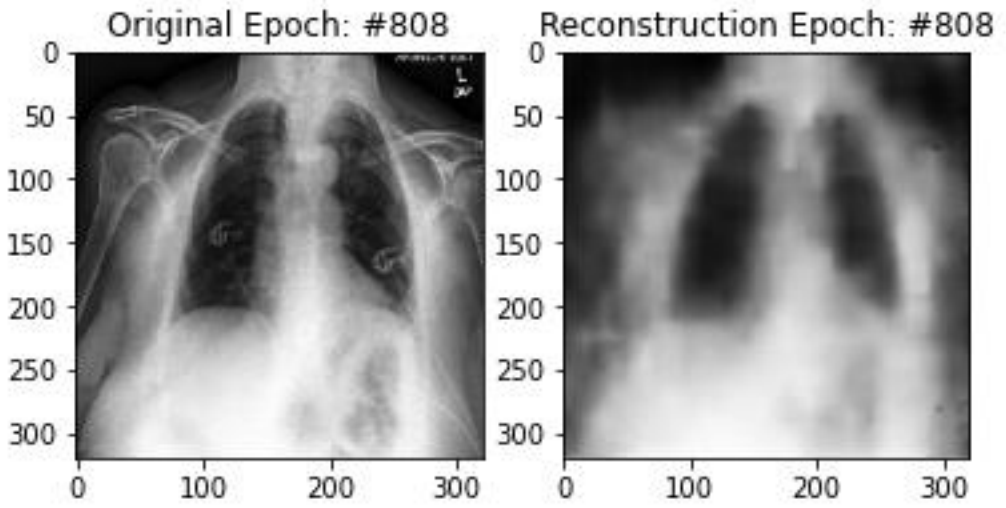


Figure 2A

The goal of training is to have a trained VAE be able to generate a random image of a chest x-ray based on what it learnt from real examples. The beauty of the VAE is their ability to generate examples that are similar to the input, but not the same. This can be useful to generate more of a type of image. For instance, VAE's can especially be useful for graphics designers who may need to generate more examples of houses for their game. It takes the hard work out of having to develop those houses themselves by hand. However, for this research, its purpose is to produce an image of a chest x-ray that is similar enough to some already existing chest x-ray that has

been diagnosed by a medical professional. The generated example should not have the same medical diagnosis as the original since it is not an exact replica.

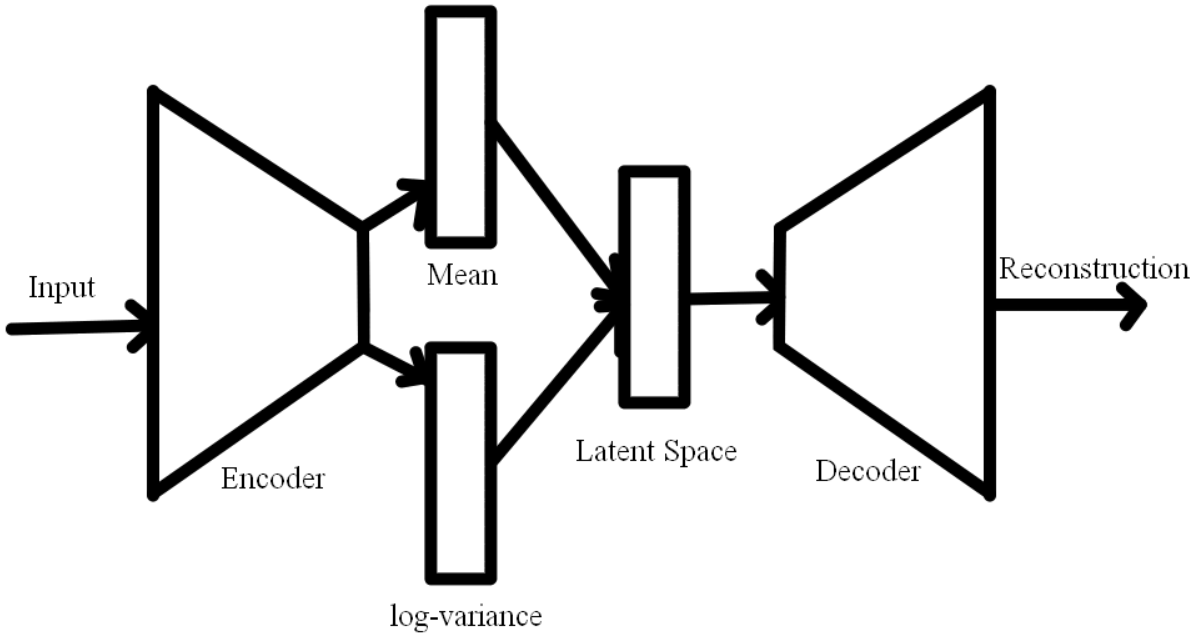
Another type of deep learning method that is similar to a VAE is a Generative Adversarial Network (GAN) (Goodfellow et al., 2014). A GAN consists of two models, one that is the generator and another that is a discriminator. Both sub models work to together so that the generator is able to produce an example that tricks the discriminator into thinking the example is real, and the discriminator tries to tell the difference, forcing the generator to be more realistic. An example of a working GAN can be seen on the website thiscatdoesnotexist, which generates the profile of a cat that "does not exist." Similar to this VAE, it was trained on real images of cat faces and learned how to generalize what a cat's face is supposed to look like and generate the example of the cat face. Before research began, there was a choice whether to use a GAN or VAE to generate the counter example. A VAE was chosen due to time constraints. In the future, it would be interesting to expand upon this research and compare generative examples of chest x-rays from both a VAE and a GAN. But for now, GANs are out of the scope of this research.

VAE IMPLEMENTATION DETAILS

The VAE for this project consists of an encoder, decoder, and latent space (Doersch, 2021)(Figure 1.2.A). During training, the VAE encodes the input using a given number of convolution layers into a condensed size. This resulting, smaller, representation of the input is said to live in the "latent space," a space not directly measurable that consists of only images of the desired type. Then using the latent space representation, the image is put through the decoder, consisting of convolution transpose layers, to generate the reconstructed image that is supposed

to be a resemblance of the original image. For this encoder, there are 7 convolutional 2D layers, with a ReLU activation function after each layer. Each layer reduces the image by a certain factor while also increasing the size of the out channels by double their amount.

Figure 2.1.A



After some experimentation, the kernel size of three with a stride no larger than two gave most promising results. As mentioned above, the larger version of the x-rays was not used. This was because using an appropriate number of layers in the encoder and decoder proved to difficult without using a larger kernel, stride, and padding. After some testing with a larger stride, the images being produced from the VAE were missing pieces in different patterns (figure 2.1.B). To save computational power by using as little layers as possible, it was decided to sacrifice the better resolution of the larger versions (2828x2320) for the smaller versions (320x320) (figure 2.1.C) which had more promising results.

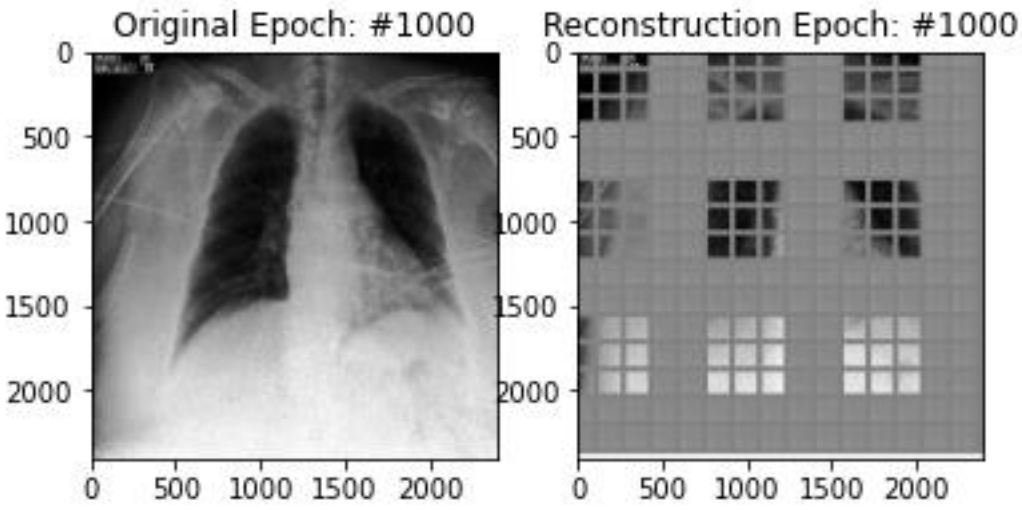


Figure 2.1.B

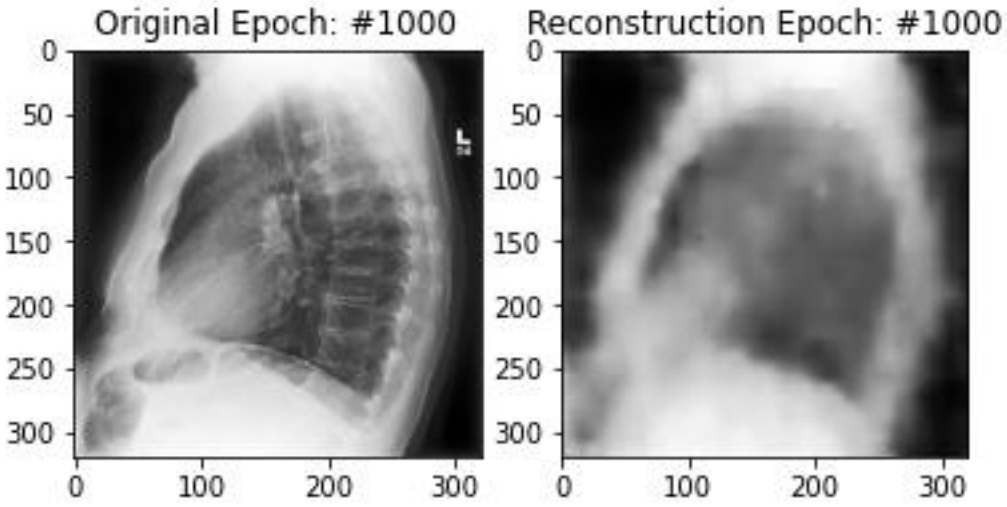


Figure 2.1.C

The role of an encoder in a VAE is to compress the input. For this research, the encoder takes in the entire image that is a multi-dimensional array that has elements to represent each of the pixel values. Pixel values can range from 0 to 1. These values represent the color of a pixel, 0 being white and 1 being black. Each chest x-ray is a 320x320 pixel image, so this incoming multidimensional array is 1x320x320. The encoder is made up of multiple convolutional layers that compress the image so it can be used to sample from in a one-dimensional space. The goal of each convolution layer in the encoder is to compress or down-size the image some and then put that compressed layer through an ReLU activation function. For this encoder, each layer compresses the image by a factor of two (figure 2.1.D). At the end of encoding, we are given an array that is of size 512.

```
#[1, 320, 320]
self.encoder = nn.Sequential(
    nn.Conv2d(1, 8, 3, stride=2, padding=1), #8, 160, 160
    nn.ReLU(),
    nn.Conv2d(8, 16, 3, stride=2, padding=1), #16, 80, 80
    nn.ReLU(),
    nn.Conv2d(16, 32, 3, stride=2, padding=1), #32, 40, 40
    nn.ReLU(),
    nn.Conv2d(32, 64, 3, stride=2, padding=1), #64, 20, 20
    nn.ReLU(),
    nn.Conv2d(64, 128, 3, stride=2, padding=1), #128, 10, 10
    nn.ReLU(),
    nn.Conv2d(128, 256, 3, stride=2, padding=1), #256, 5, 5
    nn.ReLU(),
    nn.Conv2d(256, 512, 5), #512, 1, 1
    nn.ReLU(),
    Compress()
    )
```

Figure 2.1.D

After encoding the input into an array, two linear layers are used to produce vectors representing the mean (mu) and log-variance of the distribution of the point in latent space. Before the output from the encoder is used for the mean and log-variance, it must be put through

a linear layer with a ReLU activation function. The latent space is the hidden representation of the input that is efficiently compressed. For the latent space (figure 2.1.E), it takes in both the mean and log-variance and uses that to generalize the latent space. Once the latent space is computed, it must go through a linear layer and ReLU activation function before going through the decoder.

$$Latent\ Space = mean + (e^{.5logvar} * \varepsilon)$$ (Figure 2.1.E)

Where ε is a randomly sampled multidimensional Gaussian random variable with mean zero and standard deviation of 1

After developing the latent space, a decoder is necessary to generate probability distributions to the image it outputs. The role of the decoder is to use the latent space to then generalize what an image from the dataset or input should look like by up-sampling from the latent space. Each element the vector outputted by the decoder represents the pixel value of the image it is trying to generate from the latent space. The decoder takes in the latent space vector, dubbed z. The decoder is made up of multiple convolutional transpose layers that decompress the image to get an array that is the same size as the original input 1x320x320. The goal of each convolutional transpose layer is to upscale the dimension of the vector by two and put that vector through a ReLU activation function (figure 2.1.F). At the end of decoding, we are given a multi-dimensional array that is the same size of the original image. This array can then be converted into an image.

```
#[512, 1, 1]
self.decoder = nn.Sequential(
    Decompress(),
    nn.ConvTranspose2d(512, 256, 5), #256, 5, 5
    nn.ReLU(),
    nn.ConvTranspose2d(256, 128, 3, stride=2, padding=1, output_padding=1), #128, 10, 10
    nn.ReLU(),
    nn.ConvTranspose2d(128, 64, 3, stride=2, padding=1, output_padding=1), #64, 20, 20
    nn.ReLU(),
    nn.ConvTranspose2d(64, 32, 3, stride=2, padding=1, output_padding=1), #32, 40, 40
    nn.ReLU(),
    nn.ConvTranspose2d(32, 16, 3, stride=2, padding=1, output_padding=1), #16, 80, 80
    nn.ReLU(),
    nn.ConvTranspose2d(16, 8, 3, stride=2, padding=1, output_padding=1), #8, 160, 160
    nn.ReLU(),
    nn.ConvTranspose2d(8, 1, 3, stride=2, padding=1, output_padding=1), #1, 320, 320
    nn.Sigmoid()
    )
```

Figure 2.1.F

With the loss function, the goal was to maximize Evidence Lower Bound (ELBO). ELBO is the lower bound of the posterior log-likelihood. For this type of VAE, the EBLO consists of two parts, the log-likelihood corresponding to the reconstruction error and the Kullback-Liebler (KL divergence) (figure 2.1.G). The KL Divergence is there to keep the representations of the latent space diverse while also preventing images with the same representation from being considered different representations. For the actual loss, Mean Squared Error Loss (MSE) was utilized. It helps with measuring how much information is lost from the input when trying to compute the reconstruction. MSE is useful for regression because larger errors are more penalized than smaller ones. It is also useful for the convenience of computing the gradient.

$$ELBO = MSE(y_{recon} - y_{input}) - \frac{1}{2}\sum 1 + logvar - mean^2 - e^{logvar}$$ (Figure 2.1.G)

To put things together, during training, all examples in the current partition for the dataset are put through the VAE. There, each example is put through the encoder to condense the image down. The log-variance and mean are sampled from the encoder to develop the latent space. Then, the decoder uses the latent space to generalize what the input image should have looked like in the form of a reconstruction image. The training of the VAE proceeds through gradient

descent, a method to minimize a function. In this case, the function is the loss function, which measures the reconstruction compared to the input image. Gradient descent can adjust the weights (or filters) in the convolutional layers to achieve the best loss.

Once training was completed, the model could then use the latent space and the decoder to generate an example that was made up but looked like it could have come from the original dataset, which is the goal of a VAE. In order to develop a sample from the latent space without it being altered or affected in a bad way, both the mean and log variance are set to zero, thus drawing a sample from a zero-mean unit-variance normal distribution. This distribution is what the KL divergence term compares against, so the true examples have this distribution in the latent space. Afterwards, the latent space and decoder are running the same as when training. The decoder is supposed to output an image that resembled an image of a chest x-ray, but some tests proved that that was not always the case. Sometimes a random chest (figure 2.1.H), random noise (figure 2.1.I), or a default generic chest x-ray (figure 2.1.J) were generated.

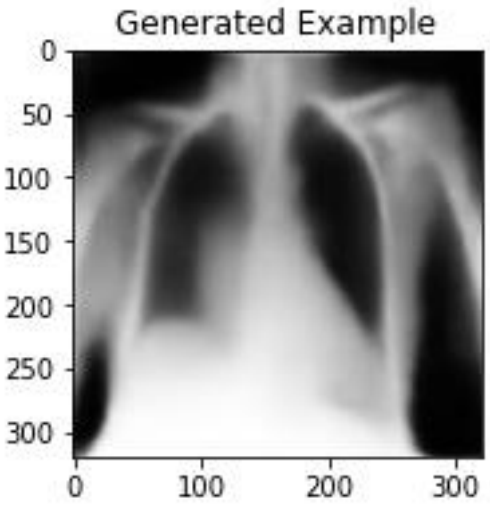


Figure 2.1.H

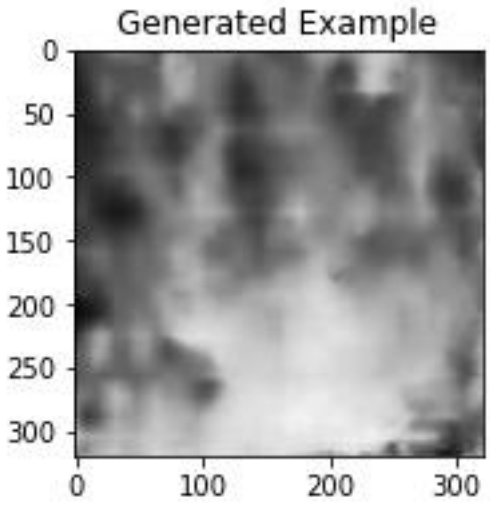
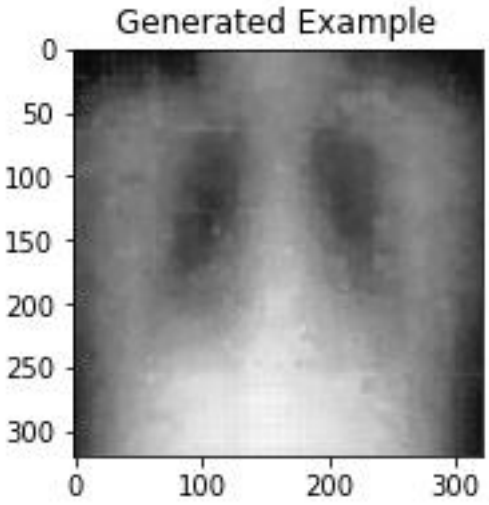


Figure 2.1.I

Figure 2.1.J



INITIAL TESTING

The goal for the VAE is for it to be able to reconstruct the input well during training so it appears to be mimicking the input. The other goal is for the VAE is able to generalize the latent space, so the model is capable of generating a synthetic image of a chest x-ray when trying to generate a convincing random example. We optimized the VAE multiple times with different versions in order to achieve this goal. During each optimization change, the trained model was gauged for both its mimicking ability during reconstruction and its quality generated examples.

For testing purposes, each version of the VAE was tested at least 5 times. The results are then compared to one another so that the results are ensured to be consistent such as the gradient descent of the random weights leading to different optima. Random sampling is also not required because the randomness was already set by the VAE during training. There were also hyperparameters that were altered and tested. These hyperparameters include the batch size, number of epochs, and the size of the dataset. The batch size represents how many examples are put through the model at a time during training before the weights and other inner parameters are updated. So, for instance, a batch size of 1 only has one example in the model at a time, whereas a batch size of 32 has 32 examples put through the VAE at a time. Usually, a higher batch size would increase the overall speedup of training the model. The number of epochs represents how many times the model goes through forward and back propagation with all the examples during training. With each iteration, the model learns more about the input. Controlling the number of epochs is crucial so the model's parameters are not underfit or overfit. Underfit means the model has not learned enough about the input to be able to generate an appropriate output. Overfit is where the model's parameters are so tuned with the data that it basically memorized the input

and is only outputting what it memorized. They were not considered to be version changes since they did not affect the results or method of the VAE to warrant such a change.

With batch size, sizes of 1, 8, 16, and 32 were tested. It was found that having a batch size of 1 or 8 performed faster overall over batch sizes of 16 and 32. The question for the number of epochs was how much was too much? We varied the tests from being 100 to 1000 epochs. It was difficult to gauge whether the loss function was converging because early on during training, the loss bounces in a range of certain values but it never seemed to decrease. It is around the 25-epoch range where the VAE began to produce a mimicking reconstruction during the training phase. After that stage, the number of epochs determined the quality of the generated example. With 100 epochs, the example produced random noise with the validation partition and an image that had some aspects but not all with the training partition. To help speed up the testing process, the validation partition of the dataset was used so that results could be immediate. 100 epochs take 20 minutes to train on the validation set, whereas it takes 10 hours to complete with the training partition. Once the results were deemed to be working, the testing partition of the dataset was used to generate the final trained model.

It was also discovered that multiple restarts were required. Some tests would have the VAE produce a reconstruction of the input x-ray, that perfectly mimicked the input x-ray during training. With other tests, the VAE was not capable of mimicking the input. With these poorer-quality trained models, the generated example was identical the reconstruction image produced by the VAE during training. The mimicking VAE produced a generated example that varied every time, with some looking not of human origin and some sharing the same characteristics of a human chest x-ray. The main goal of the trained model is for it to be able to reconstruct well (figure 2.1.C) and be able to produce a human chest-related generated example (figure 2.1.H).

Once the initial version of the VAE was completed and was deemed to be able to reconstruct the original input and produce a generated example that was satisfactory and in the scope of a human chest x-ray, it was decided the results were not accurate enough. With training a smaller partition, about 10000 examples of the dataset for 1000 epochs, the VAE can replicate the input chest x-rays during reconstruction and generate a decent chest x-ray that can be recognized to be a chest x-ray (figure 2.1.C). The problem was that the generated chest x-rays were not detailed enough (figure 2.1.H). They were missing the ribs and other crucial details. This caused us to explore changes to both the loss function and methods of training the VAE.

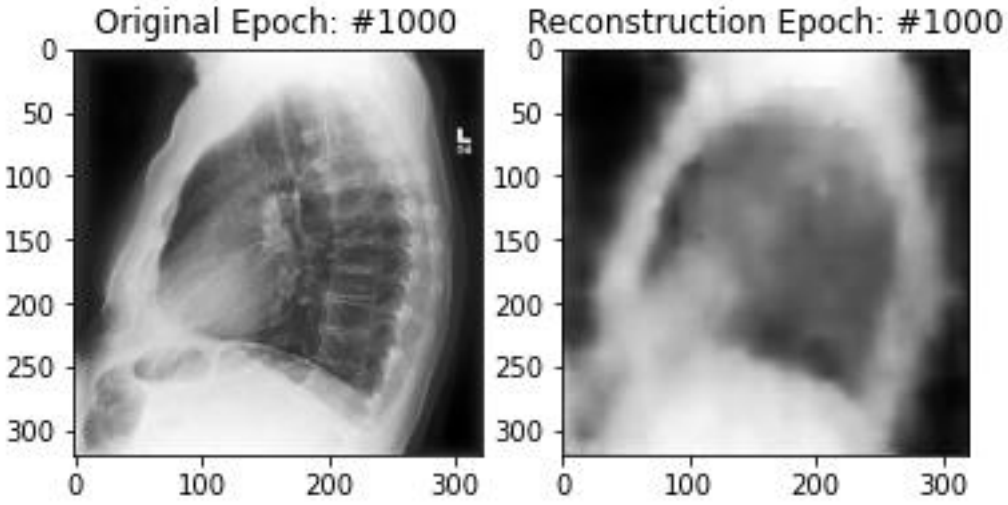


Figure 2.1.C

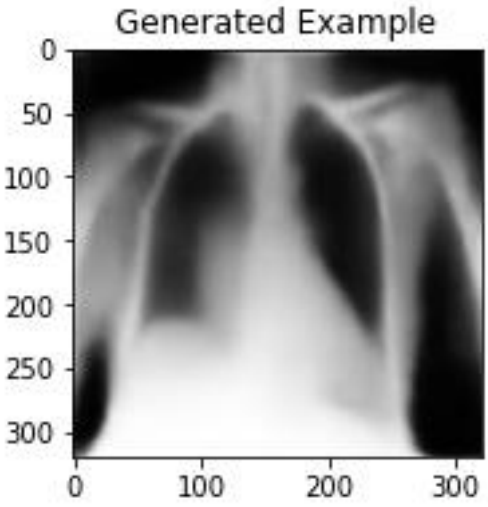


Figure 2.1.H

OPTIMIZATIONS

After training the first version of the VAE, we observed the loss values during training. After a certain number of epochs, the loss values began to converge or stay around a certain range. This was troublesome considering the model was able to produce both a mimicking reconstruction and a somewhat adequate generated example.

In the quest to obtain better generated examples, the research took a turn that involved changing the initial loss function. Asperti and Trentin had addressed the difficulty in finding the right balance between the loss and the KL divergence (2020). Their solution was to have a hyper parameter, Gamma, that could be used to control the balance between the KL-divergence and reconstruction loss terms of the ELBO. They chose to update gamma by finding the minimum between gamma and 99% of gamma and 1% of the loss added together (figure 4A). Gamma is useful for limiting how much the loss should affect the KL divergence (figure 4B).

$$Gamma = min(gamma, (.99 * gamma + .1 * loss))$$ (figure 4A)

$$ELBO = MSE(y_{recon} - y_{input}) - \frac{1}{2}\sum 1 + logvar - mean^2 - e^{logvar}$$ (figure 2.1.G)

$$ELBO = \frac{1}{2*gamma} MSE(y_{recon} - y_{input}) - \frac{1}{2}\sum 1 + logvar - mean^2 - e^{logvar}$$ (figure 4B)

With using Gamma, we found that the model did not mimic during reconstruction and the KL term stayed stagnant (figure 4C). All the reconstruction images resembled the same generic chest x-ray image (figure 4D).

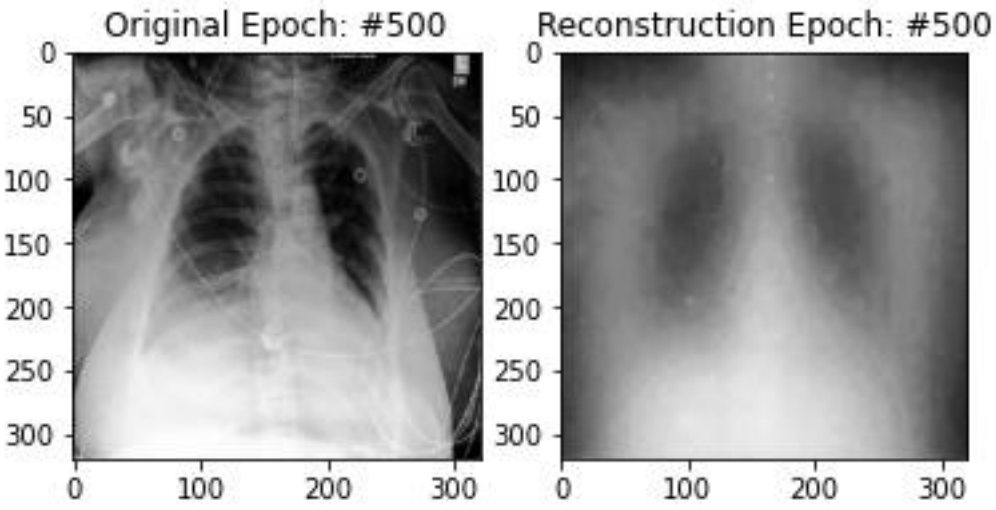


Figure 4C

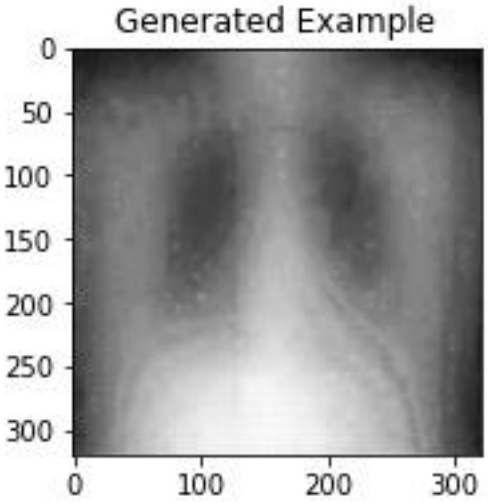


Figure 4D

Since the KL divergence was showing to not affect the loss, it was decided to run some ablative studies where the KL Divergence term was not included with MSE loss. This is to gauge the importance of the KL Divergence in the loss function. After doing so, the model started mimicking during training (figure 4E), but the generated example did not resemble a chest at all (figure 4F). This could be because the KL Divergence's purpose is to promote generalization.

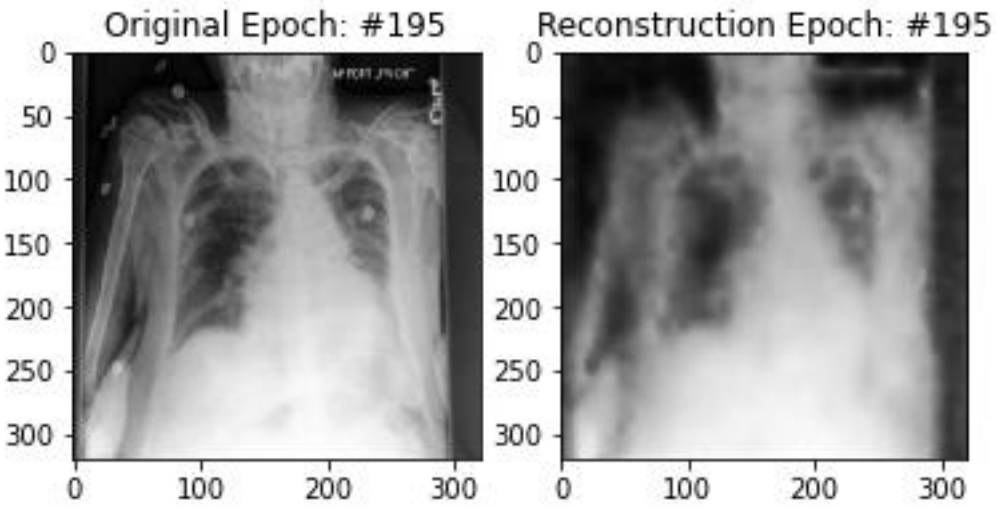


Figure 4E

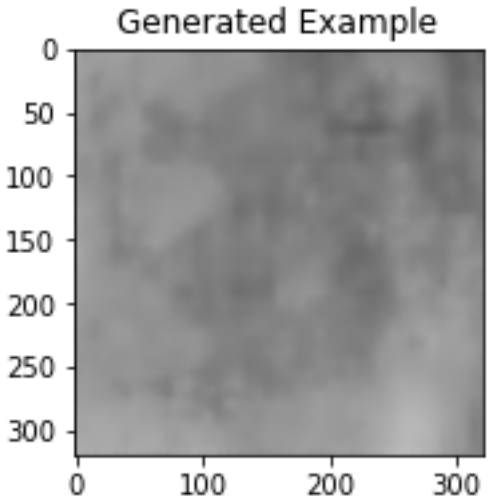


Figure 4F

Generally, a KL divergence term is required in a VAE. So, it was decided to test the loss function with gamma but without the KL divergence term for the first few epochs and then introduce it later on. KL divergence was introduced during the 25th epoch because of previous test without KL showed that it stopped improving around there. For the epochs without KL, the model was mimicking during the reconstruction. However, after it was introduced, the model

began outputting the generic chest x-ray image seen before during reconstruction. The generated example resembled the same generic chest x-ray as well.

After multiple optimizations and tests, we found that the best results were obtained from the original version of the VAE. The optimizations such as utilizing Gamma in the loss and the introducing of the KL divergence proved to not help with both the reconstruction and the generated example. Most optimizations only allowed the model to generate a generic image of a chest x-ray during the reconstruction and for the generated example since the model converged early on, so the model assumed that was what an image of a chest x-ray should look like. This was evident with the values of the loss. For now, the best that can be obtained from this VAE is a generated example that is of resemblance to a human chest but is missing details such as the ribs. The blurriness and the missing details of the VAE is not uncommon. This has been seen with other VAEs. A solution to this could be to consider the details on a separate layer on top of the VAE's generated example.

WHAT'S AHEAD

The VAE developed during this research can be expanded upon to be used in the explanation of a model's diagnosing accuracy and the reason for its diagnosis over another. The usage of a VAE allows for a visual representation that can be easily understandably by a human and someone who is not well-versed with machine learning. As explained above, the original input and the generated counter example from the VAE can be used with a model train to diagnose. Both of their diagnoses can then be compared over one another to give an understanding where on the generated example is the model focusing on over the original to give its diagnosis. This explanation can best be implemented using the CDeepEx method described in (Feghahati, et al., 2020).

CONCLUSION

This research seeks to provide a way for other researchers to be able to prove the validity of a Deep Neural Network trained to diagnose patients based of images of the x-rays. In order to provide this method of analysis, we utilize a Variational Autoencoder to generate false examples of said x-rays to then be compared with an actual x-ray when given as input in a model pretrained to diagnose x-rays. To train the VAE and verify that is able to generate an anatomically correct false example of an x-ray, we make use of a publicly available dataset, Chexpert, that is made up of real images of human chest x-rays that are diagnosed and verified by medical professionals. A smaller, condensed version of the dataset is utilized for computation and time constraints. It was found that both versions of the dataset, both the original and the condensed versions were capable of providing the same results.

When building the initial version of the Variational Autoencoder, it was made up of an encoder, latent space, and decoder. The encoder condenses the original input into a form that can then be sampled from to develop the latent space. The latent space is then used by the decoder to generalize the latent space to generate the counterfactual example of an x-ray image. The loss function was simply the Mean Squared Error loss with KL divergence as the regularizer. The loss function tries to minimize the difference between the original image and the reconstruction image. Based off the original structure of the VAE, some basic tests were established to determine what would cause the optimal results with both the loss, reconstruction image, and generated counter example. First, the number of epochs had to be chosen. For testing purposes, a smaller partition of the dataset of about ten thousand examples were used. For this partition, it was found that Variationa Autoencoder started mimicking for the reconstruction at around epoch twenty, depending on if the stars were aligned. This was the reason for multiple restarts with

training the model. Sometimes the model was capable of mimicking during training and sometimes it did not. We only considered runs with the model where it did mimic during reconstruction. It was found that model did consistent reconstructions around epoch 100 but the generated example was not of human origin, which is not usable for the requirements of this research. In the end, a model trained over 1000 epochs gave the best generated examples which resembled a human chest x-ray, except it was missing some crucial details such as the ribs and the image was overall blurry, which is not uncommon for a Variational Autoencoder. Since the main goal of this research was to generate a convincing enough generated counter example that could rival a real x-ray and fool another model's diagnosing strategies, this research took the turn of optimizing the Variational Autoencoder. For optimizations, we tested the model with the usage of Gamma and introducing the KL Divergence term. We found that the model did not perform better than its original structure before the optimization. The best we could produce as examples were chest x-rays that were slightly blurry and missing the ribs. This could be improved later on by developing a method to add on the details as an extra layer on top of the generated example.

Although this research does not provide the explanation necessary for humans to have a better understanding of A.I, it still provides the counterexample required for developing the explanation. In conjunction with CDeepEx, this VAE is useful for providing a human understandable explanation to a model's diagnosis as well as proving whether or not the model is behaving correctly. The use of a visual explanation is beneficial for humans to understand and trust the usage of A.I. as a medical diagnosis tool. Having A.I in the medical field is beneficial

for providing accurate diagnoses which are not plagued by human error and for expanding the resources necessary to help more patients.

REFERENCES

Amir Feghahati, Christian R. Shelton, Michael J. Pazzani, and Kevin Tang (2020). "CDeepEx: Contrastive Deep Explanations." European Conference on Artificial Intelligence.

Andrea Asperti and Matteo Trentin (2020). "Balancing reconstruction error and Kullback-Leibler divergence in Variational Autoencoders." arXiv. https://arxiv.org/abs/2002.07514

Doersch, Carl. (2016). Tutorial on Variational Autoencoders.

Diederik P. Kingma and Max Welling. *Auto-encoding variational Bayes*. International Conference on Learning Representations (ICLR), 2014.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural 67 Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014. URL http: //papers.nips.cc/paper/5423-generative-adversarial-nets.pdf.

Michael Pazzani, Amir Feghahati, Christian Shelton, and Aaron Seitz (2018). "Explaining Contrasting Categories." IUI Workshop on Explainable Smart Systems.