Investigating Latent Space of Variational AutoEncoders (VAEs)

Vedant Devank Patadia
Lakehead University - Computer Science
email: vpatadia@lakehead.ca

***Abstract*— Latent space representations are the building blocks of any machine learning or deep learning structure through which one can make out the general working of any newly made model. But it is only due to the experience with building and training countless structures that an engineer can predict the consequences of various scenarios that the structure may get into which is generally termed as data intuition. Data intuition is nothing but simply learning a trend in how the latent space behaves as the model is trained, i.e., practically delving into the dependency between the different inputs and the outputs being generated through the training process. The algorithms relying on the in depth understanding of latent learning are coming into the trend as the computational capacity is increasing to the new heights. This project is geared towards Understanding one such Deep Learning Network, that is, Variational AutoEncoders using three tiers of datasets with increasing complexity, the overall complexity is low overall but as is the Complexity of the Deep Learning model being used.**

***Index Terms*— autoencoders, deep-learning, latent-space, vomplexity-variation, interpretability**

I. INTRODUCTION

Latent space is a very abstract term, and it has been in the play ever since we have been embedding data in the smaller compressed forms, some references in the field of probability and statistics even go as early as 1959 by Anderson, T. W [1].

As described in [2], latent space is a ”reduced-dimensionality vector space embeddings of data, fit via machine learning”. Which also means, vector embeddings of data, where the distance between similar vector is minimized.

The term latent space is quite broad, and is literally relevant to almost every aspect of machine learning, let alone deep learning. Thus, The review is particularly focusing on how the analysis of latent space helps in understanding models as well as the methods or models that employ the latent space/variables at their very core algorithm.

The basis of latent space representation is that, we embed these higher dimensions within lower dimensional representation, or typically compressed form which is typically a lossy process. In a more specific sense, the higher dimensional data can be learnt by a lower dimension latent space, which serves as a manifold for further generation, as described in [3].

A. Related Work

As described in [4], latent space is a ”reduced-dimensionality vector space embeddings of data, fit via machine learning”. Which also means, vector embeddings of data, where the distance between similar vector is minimized.

The term latent space is quite broad, and is literally relevant to almost every aspect of machine learning, let alone deep learning. Thus, The review is particularly focusing on how the analysis of latent space helps in understanding models as well as the methods or models that employ the latent space/variables at their very core algorithm.

Further as inspired by [5], the model being termed as a “Black Box”, is simply not acceptable at certain tasks as reaching saturation on such critical tasks is not a simple task and what more is that the “Black Box” representation of AI models may lead to them being never leaving the Research phase off itself. What more is that the research in this field is never ending, and the importance is often overlooked, as surveyed in [6].

The importance of being able to observe the performance of an AI Model is non-trivial, and sometimes able to solve identified problems in short period that otherwise may have taken longer. This is quite evident from the development of StyleGAN2 from [7], where the authors not only did improve the model capacity but also solved artifact problems that were inherent in StyleGAN[8]. Although its a good example involving interpreting, incorporating and visualizing the representations for which the latent space data is responsible for, but the research is quite a high profile one involving researchers backed by NVIDIA.

Autoencoders and GANs are quite literally playgrounds based off on Latent Space of a Deep Learning model. Where in Autoencoder, the model compresses the data into a latent space representation and then extrapolates the latent space back into data and in a Generative Adversarial Network, a Generator and Discriminator(or Classifier) pass on data back and forth, usually through the latent space in-order to train the generator on the problem at hand.

Like in, “Parallel WaveGAN: A Fast Waveform Generation Model Based on Generative Adversarial Networks with Multi-Resolution Spectrogram [9]” The authors train a GAN to generate spectrograms based on the adversarial input function provided to it, practically making a text to speech network.

The WaveGAN, is accurate an fast but needs quite the training data and time which is the general rule of thumb for training GANs, they require data and time to train for accuracy to prevail, but the authors of the publication ”Training generative adversarial networks with limited data. [10]” thought differently, while also dealing with the discriminator overfitting, by attaching an adaptive non-leaking invertible augmentation pipeline to the generator and discriminator,

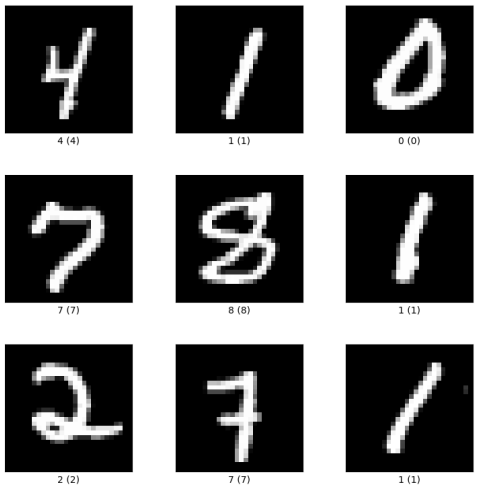

Fig. 1. MNIST Handwritten Digits Examples. Image credits, Tensorflow Dataset Catalog

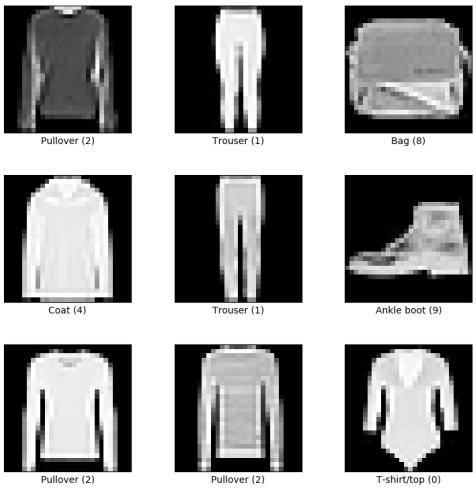

Fig. 2. MNIST Fashion Examples. Image credits, Tensorflow Dataset Catalog

i.e., on the latent space itself and they were able to control divergence and stablize the training process through it.

While adaptive augmentation is quite novel, StyleGANs are currently heated to crisp how else would you describe a network that added more noise to remove noise. Well not exactly, but the StyleGAN Proposed in "A Style-Based Generator Architecture for Generative Adversarial Networks [8]", is able to take in noise from one end and then result in detail on the other, practically merging the inputs, with the noise being described as style and the phenomenon called style mixing using latent codes.

Under the mist of StyleGAN lies the concept called, "Self-Attention Generative Adversarial Networks [12]", SAGAN is said to be the base of StyleGAN where self attention simply means drawing out context based off on the models own output, this enables long range dependency in attention driven tasks to generate images and practically beats the state of the art GAN models while doing so.

It just shows how much uncharted territory the latent space, the space of infinite dimensionality has towards it and it is not just GANs, Autoencoders are quite the show as well like, "3D MRI brain tumor segmentation using autoencoder regularization. [13]" where the autoencoder is used as an regularization mechanism in training the CNN, Although the CNN itself is structured as an encoder-decoder network, taking in scans and bringing out tumor locations. The question proposed here and the experimental approach is quite similar to [14], where the authors use Generative Adversarial Networks[14] in interpreting the latent space of the same. The application itself is what they call InterFaceGAN, in which they explain the Latent Space of GANs as a Riemannian manifold and their modification of popular GANs like StyleGAN[8] and PGGAN[15] to encode semantics into them, i.e., add conditional manipulation on

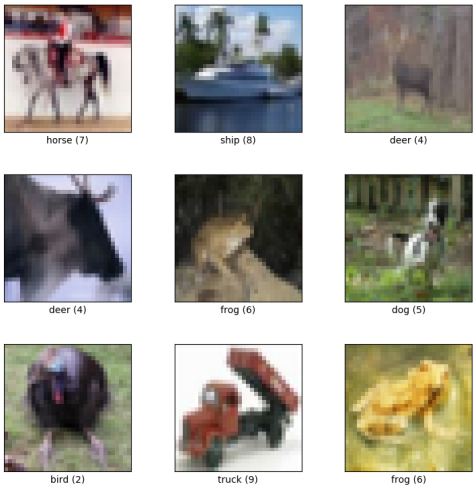

Fig. 3. CIFAR-10 Dataset. Image credits, Tensorflow Dataset Catalog

the GAN to create different set of manifolds on the same image based off on an attribute. Their interpretation in latent space is mediated by change in the input Faces by attribute manipulation, where they find that the more borderline the generation is, the more lookalike output will be to the input to the model, and if they go farther then at a certain point the faces will come out to be entirely different.

The expectation is quite similar in this paper, but instead of state-of-the-art models, highly complex images and large number of classes as in the development of StyleGAN[8], the basis of network models proposed here is Variational Autoencoders that make the foundations for the higher level

Fig. 4. MNIST Handwritten Digits Predictions.

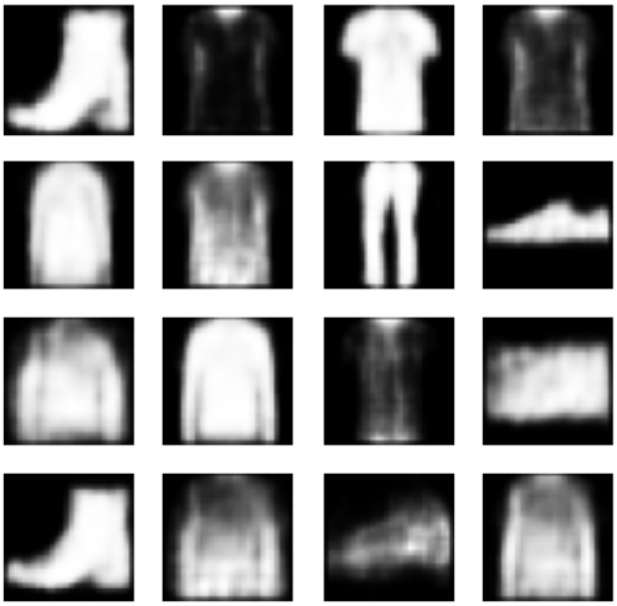

Fig. 5. MNIST Fashion Predictions.

architecture of models such as GANs[16]. Further, how does the latent space behave on variations in data complexity while keeping the model complexity similar in each iteration. Such research may lead to interesting, visuals where we will be able to see the interpolations that the model learn and how intermediate visuals differ between different complexity.

II. METHODS

The general idea here is to,

1) Compile model, so that we can run tests on it
2) Train on a dataset from the chosen list, so the model can learn. Here, I have chosen 10 epochs which was set by default in the Reference followed here.
3) Sample from the dataset to generate a prediction matrix
4) Create a prediction matrix, from the each epoch
5) Generate a GIF animation from the prediction matrices, this will help visualize the model training flow
6) Get the latent space from the samples
7) Plot the Latent space on the matrix

Adding to it, The three levels of complexity i have chosen here are,

- Black and white Images, MNIST Handwritten Digits [17]. Here the images contain, simple edges, lines and curves where the blob sizes are not that concentrated and the Differences can be made out by simple Machine Learning Algorithms as seen in Figure 1
- Black and white Images, MNIST Fashion Dataset [18] Here the images are quite similar to Original MNIST Digits, but the blob sizes and concentration of object pixels are aggregated also now there are complex shapes and texture to the objects as seen in Figure 2.
- Color Images, CIFAR-10 Dataset [19] Here the images are in Color, the tests were performed considering both RGB format and Grayscale format with complex image blobs and shapes as seen in Figure 3. Here the dataset is more real-world based and now I expect the hidden dimensionality will affect the results.

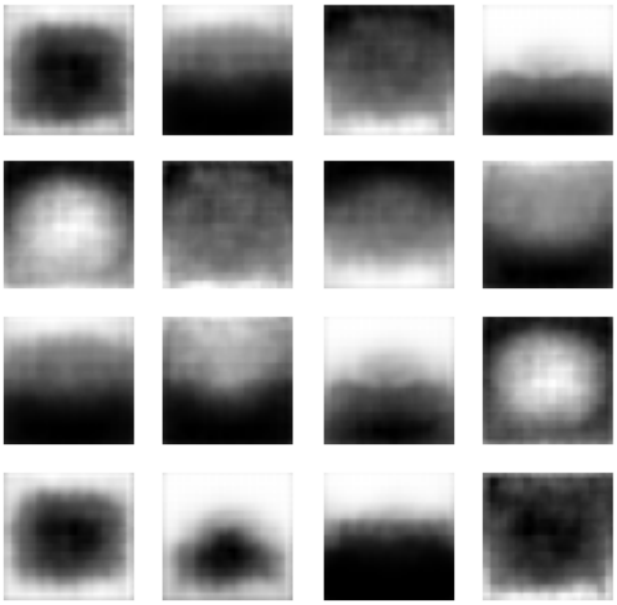

Fig. 6. CIFAR-10 Predictions.

III. RESULT

The predictions on MNIST Handwritten digits as shown in Figure 4 and on MNIST Fashion shown in Figure 5 are crisp and fairly clear as running the training paradigm for only 10 epochs.

In addition, the results in Figure 4 and Figure 5 are slightly distorted but the pixelation in original Figure 1 and

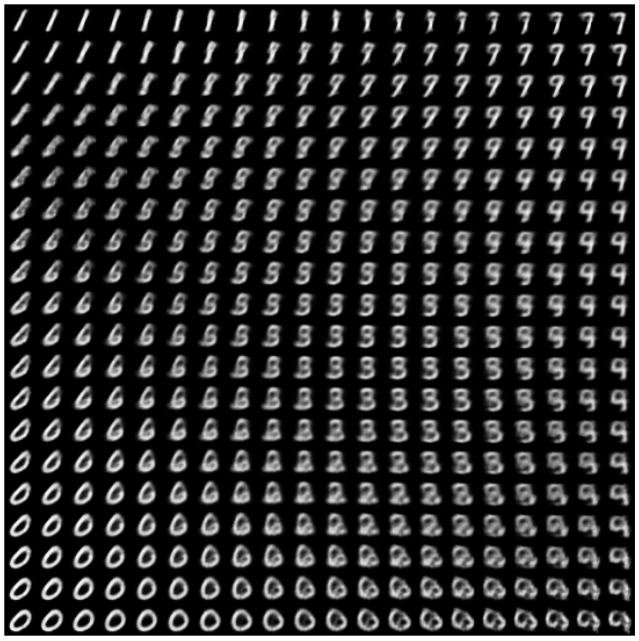

Fig. 7. MNIST Handwritten Digits Latent Space.

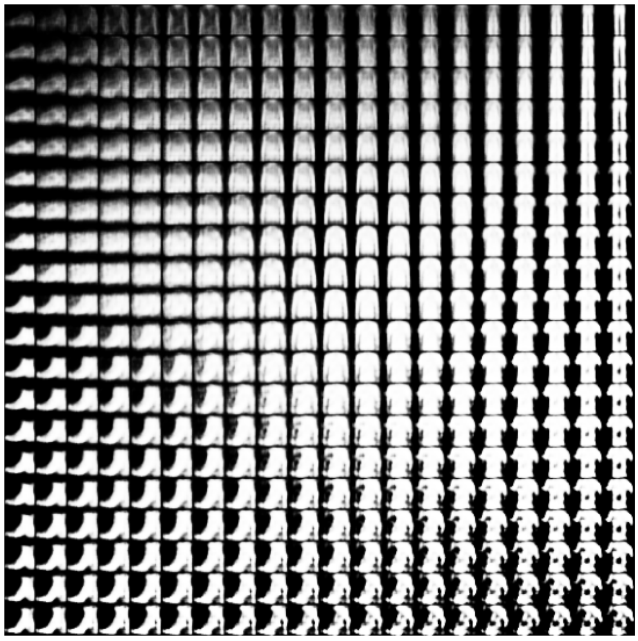

Fig. 8. MNIST Fashion Latent Space.

2 is reduced heavily Also, looking at the prediction on the CIFAR-10 dataset in Figure 6 We are able to see that,

- There are some round objects
- Some objects have low intensity background and some have high intensity ones
- Some objects are darker and some of them are light
- The contrast is seen but the texture and precision is lost

The latent space follows the same trends as in the predictions, the reason is that it is almost the same inference that is carried into the predictions.

In each section of Figure 7 and Figure 8 we can see simple interpolations between different classes and this is what we call the latent space learning the image interpolation between different classes as described in [20]

As for CIFAR-10, Figure 9 is concerned the data is quite fuzzy as is the images in the dataset, we can see slights of each classes in it but this can simply be a case of underfitting.

IV. DISCUSSION

The results follow the pattern that was set out by the Visualization in the paper on Auto-Encoding Variational Bayes in [21] which were generated using 2D Latent Space.

In, MNIST Handwritten Predictions in Figure 4, We are able to see some overlap in classes,

- Digit Class 6 and Digit Class 0
- Digit Class 9 and Digit Class 7
- Digit Class 6 and Digit Class 8
- Digit Class 0 and Digit Class 8

In, MNIST Fashion Predictions in Figure 5, We are able to see that

- Texture is almost lost
- There are Some Structural Distortions
- Some overlap in classes can be made out but not as concretely as previously

Finally, moving on to our final dataset we are beginning to see some problems with our implementation. As we can see in Figure 6 the results are almost uninterpretable, further the structural integrity and texture is lost. We can quite literally make out nothing.

V. CONCLUSION

The results, or more precisely the interpretability of the latent space degrades as we feed more complex data into the same model

There can be multiple reasons for this, but here 3 major reasons do specify their importance and simplicity,

- Higher Complexity on a model with lower depth
- Low quality and resolution of images
- Inference on images converted from RGB to Grayscale
- The latent space is actually tad bit more complex to be plotted using simple methods

And the difference due to such qualities is seen thoroughly in the results of CIFAR-10 dataset.

A. FUTURE WORK

There are three typical resolutions can be made on the models,

- Quantify the channels for RGB images, so we can get more information
- Use a model with higher depth and hyperparameter optimization
- Take up more complex datasets with higher resolution, but this will require extensive hardware

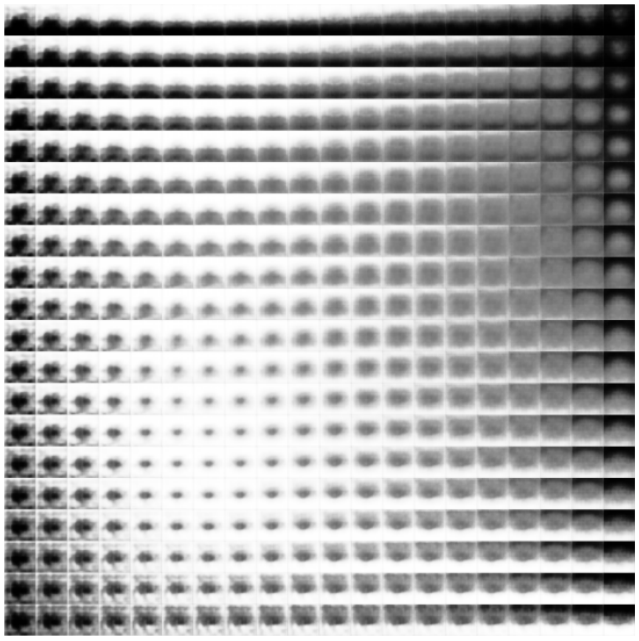

Fig. 9. CIFAR-10 Latent Space.

VI. ACKNOWLEDGEMENT

I would like to thank Dr. Trevor Tomesh for his intense help in the project, and his teaching us the Course Research Methodology which has played a huge role in the completion of this project.

I would also like to acknowledge the time and effort put in by the Marker for assessing the project. Thank you for reading the paper.

REFERENCES

[1] Grenander, Ulf. Probability and statistics: the Harald Cramér volume. Almqvist & Wiksell, 1959.

[2] Y. Liu, E. Jun, Q. Li, and J. Heer, "Latent Space Cartography: Visual Analysis of Vector Space Embeddings," Computer Graphics Forum, vol. 38, no. 3, pp. 67–78, 2019, doi: 10.1111/cgf.13672

[3] G. O. Young, Synthetic structure of industrial plastics (Book style with paper title and editor), in Plastics, 2nd ed. vol. 3, J. Peters, Ed. New York: McGraw-Hill, 1964, pp. 15 64.

[4] Y. Liu, E. Jun, Q. Li, and J. Heer, "Latent Space Cartography: Visual Analysis of Vector Space Embeddings," Computer Graphics Forum, vol. 38, no. 3, pp. 67–78, 2019, doi: 10.1111/cgf.13672

[5] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, Electron spectroscopy studies on magneto-optical media and plastic substrate interfaces(Translation Journals style), IEEE Transl. J. Magn.Jpn., vol. 2, Aug. 1987, pp. 740 741 [Dig. 9th Annu. Conf. Magnetics Japan, 1982, p. 301].

[6] M. Young, The Techincal Writers Handbook. Mill Valley, CA: University Science, 1989.

[7] S. Chen, B. Mulgrew, and P. M. Grant, A clustering technique for digital communications channel equalization using radial basis function networks, IEEE Trans. Neural Networks, vol. 4, pp. 570 578, July 1993.

[8] R. W. Lucky, Automatic equalization for digital communication, Bell Syst. Tech. J., vol. 44, no. 4, pp. 547 588, Apr. 1965.

[9] R. Yamamoto, E. Song and J. -M. Kim, "Parallel Wavegan: A Fast Waveform Generation Model Based on Generative Adversarial Networks with Multi-Resolution Spectrogram," ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2020, pp. 6199-6203, doi: 10.1109/ICASSP40776.2020.9053795.

[10] Karras, Tero, et al. "Training generative adversarial networks with limited data." arXiv preprint arXiv:2006.06676, 2020.

[11] T. Karras, S. Laine and T. Aila, "A Style-Based Generator Architecture for Generative Adversarial Networks," 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 4396-4405, doi: 10.1109/CVPR.2019.00453.

[12] Zhang, Han, et al. "Self-attention generative adversarial networks." International conference on machine learning. PMLR, 2019.

[13] Myronenko, Andriy. "3D MRI brain tumor segmentation using autoencoder regularization." International MICCAI Brainlesion Workshop. Springer, Cham, 2018, doi: 10.1007/978-3-030-11726-9_28.

[14] S. P. Bingulac, On the compatibility of adaptive controllers (Published Conference Proceedings style), in Proc. 4th Annu. Allerton Conf. Circuits and Systems Theory, New York, 1994, pp. 8 16.

[15] W. D. Doyle, Magnetization reversal in films with biaxial anisotropy, in 1987 Proc. INTERMAG Conf., pp. 2.2-1 2.2-6.

[16] Sainburg, Tim, et al. "Generative adversarial interpolative autoencoding: adversarial training on latent space interpolations encourage convex latent distributions." arXiv preprint arXiv:1807.06650 (2018).

[17] L. Deng, "The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web]," in IEEE Signal Processing Magazine, vol. 29, no. 6, pp. 141-142, Nov. 2012, doi: 10.1109/MSP.2012.2211477.

[18] Xiao, Han, Kashif Rasul, and Roland Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," arXiv preprint arXiv:1708.07747, 2017.

[19] H. Li, H. Liu, X. Ji, G. Li, and L. Shi, "CIFAR10-DVS: An Event-Stream Dataset for Object Classification," Front. Neurosci., vol. 11, May 2017, doi: 10.3389/fnins.2017.00309.

[20] S. Mukherjee, H. Asnani, E. Lin, and S. Kannan, "ClusterGAN: Latent Space Clustering in Generative Adversarial Networks," AAAI, vol. 33, pp. 4610–4617, Jul. 2019, doi: 10.1609/aaai.v33i01.33014610. [Online]. Available: http://dx.doi.org/10.1609/aaai.v33i01.33014610

[21] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," Machine Learning, May 2014.

SUPPORTING MATERIAL

THE CVAE MODEL implementation used in the project

```
## @package modules
# Convolutional variational autoencoder class
# Adapted from TensorFlow Tutorial, Convolutional Variational Autoencoders
# https://github.com/tensorflow/docs/blob/master/site/en/tutorials/generative/cvae.
   ipynb
#
# LICENSE
# @title Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
# https://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.

import tensorflow as tf
import numpy as np


## @title CVAE
# Convolutional variational autoencoder class
class CVAE(tf.keras.Model):
```

```
## constructor
def __init__(self, latent_dimen=2, inp=(28,28,1)):
   super(CVAE, self).__init__()
   #dimensions of the latent space
   self.latent_dimen = latent_dimen
   #the encoder network
   self.encoder = tf.keras.Sequential(
      [
         tf.keras.layers.InputLayer(input_shape=inp),
         tf.keras.layers.Conv2D(
            filters=32, kernel_size=3, strides=(2, 2), activation='relu'),
         tf.keras.layers.Conv2D(
            filters=64, kernel_size=3, strides=(2, 2), activation='relu'),
         tf.keras.layers.Flatten(),
         # No activation
         tf.keras.layers.Dense(latent_dimen + latent_dimen),
      ]
   )

   #the decoder network
   self.decoder = tf.keras.Sequential(
      [
         tf.keras.layers.InputLayer(input_shape=(latent_dimen,)),
         tf.keras.layers.Dense(units=inp[0]*inp[1]*2, activation=tf.nn.relu),
         tf.keras.layers.Reshape(target_shape=(inp[0]//4, inp[1]//4, 32)),
         tf.keras.layers.Conv2DTranspose(
            filters=64, kernel_size=3, strides=2, padding='same',
            activation='relu'),
```

```
            tf.keras.layers.Conv2DTranspose(
                filters=32, kernel_size=3, strides=2, padding='same',
                activation='relu'),
            # No activation
            tf.keras.layers.Conv2DTranspose(
                filters=inp[-1], kernel_size=3, strides=1, padding='same'),
        ]
    )

## function to sample back from decoder
@tf.function
def sample(self, eps=None):
    if eps is None:
        eps = tf.random.normal(shape=(100, self.latent_dimen))
    return self.decode(eps, apply_sigmoid=True)

## function to encode data onto Latent space
def encode(self, x):
    return tf.split(self.encoder(x), num_or_size_splits=2, axis=1)

def reparameterize(self, mean, logvar):
    eps = tf.random.normal(shape=mean.shape)
    return eps * tf.exp(logvar * .5) + mean

## function to decode latent space vars
def decode(self, z, apply_sigmoid=False):
    logits = self.decoder(z)
    if apply_sigmoid:
```

```
        probs = tf.sigmoid(logits)
        return probs
    return logits
```