DEEP LEARNING TESTING AT THE FEATURE LEVEL

Amol Sahebrao Rathod
Student ID: 201593889

A DISSERTATION

Submitted to

The University of Liverpool

in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Primary Supervisor: Prof. Xingyu Zhao
Secondary Supervisor: Prof. Xiaowei Huang

Abstract

The applications of Deep learning techniques and methods are increasing at a rapid rate in every sector. Deep learning also raises concerns over its reliability in important everyday applications. It is very important to learn the basics of how this deep learning techniques work. Deep learning methods are prone to adversarial attacks. In this project we present novel deep learning testing at the feature level. A generative type of model called Variational autoencoder is used to generate the test cases. An important property called the r separation property is used in the latent feature space of the Variational autoencoder. We were able to train the Variational Autoencoder on the image datasets. The r distance was measured in the latent feature space and images were generated that were within this r distance. The generated test cases shared the same ground truth. Previous standard approach in using generative models for generating randomly through sampling are either less accurate or robust. This is just a simple approach towards finding robust deep leaning classifiers. Future work can be done in this direction learning other generative models and using high computational power.

Student Declaration

I confirm that I have read and understood the University's Academic Integrity Policy.

I confirm that I have acted honestly, ethically and professionally in conduct leading to assessment for the programme of study.

I confirm that I have not copied material from another source nor committed plagiarism nor fabricated data when completing the attached piece of work.

I confirm that I have not previously presented the work or part thereof for assessment for another University of Liverpool module.

I confirm that I have not copied material from another source, nor colluded with any other student in the preparation and production of this work.

I confirm that I have not incorporated into this assignment material that has been submitted by me or any other person in support of a successful application for a degree of this or any other university or degree-awarding body.

SIGNATURE _____A.S.Rathod_____

DATE October 7, 2022

Acknowledgements

I would like to extend my gratitude towards University of Liverpool for giving me an opportunity to work on this dissertation. I would like to thank my Primary Supervisor Prof Xing Yu Zhao and Secondary Supervisor Prof Xiaowei Huang who guided me along this topic and provided valuable feedback from time to time. I would also like to thank Prof Christian Ikenmeyer(Project Co-ordinator) who provided structure and facilities from University like computer labs, library which contributed a lot in success of this dissertation.

Table of Content

Chapter 1

1. Introduction

Due to the recent progress in machine learning we know how incredibly powerful and successful it is, many different tasks that could not be solved with software before are now solvable, thanks to deep learning and convolutional networks and gradient descent all of these technologies are working really well. Until just a few years ago these technologies didn't really work. In about 2013, we started to see that deep learning achieved human level performance at a lot of different tasks. we saw that convolutional nets could recognize objects and images and score about the same as people in those benchmarks. Part of the reason that algorithms score as well as people is that people can't tell the difference between Alaskan huskies and Siberian huskies very well, but modulo the strangeness of the benchmarks deep learning caught up to about human level performance for object recognition in 2013. That same year we also saw that for object recognition applied to human faces caught up to about human level that suddenly we had computers that could recognize faces about as well as you or I could recognize faces of strangers. We also saw that computers caught up to reading photos and fonts to us. It even got to the point that we could no longer use CAPTCHAs to tell whether a user of a webpage is human or not because the convolutional network is better at reading obfuscated text than a human is. So, with this context of deep learning working well especially for computer vision and its ever-increasing use in safety and security critical applications, there is an increasing concern over its trustworthiness, reliability, accuracy and dependency. Even with all its advancement in Adversarial examples, most of the research on Image Classification, Computer vision and Object detection is focused on the pixel level. In this project, we will explore how to test Deep learning techniques at feature level based on the Generative models. VAEs are used to generating test cases, these test cases need to have a ground truth label based on the r-separation distance (Yang, et al., 2020). For each one we can query a ML model and its testing to see if the predicted label is the same as the ground truth label therefore, we will have our accuracy.

Scope

Image classification is mostly done at the pixel level and there is not many research done in the feature space. In this project we use the r separation property proposed by (Yang,et al., 2020) which states that for real world image datasets any data points with different labels should be separated by atleast a distance of 2r. We have used type of Generative model for this project. First, we build a Variational Autoencoder and after we pass aur data through it and train the model, we will find the r separation distance in the latent space of the Variational Autoencoder, and the decoder will act as a generator network where we will generate our input data from sampling through this latent space of VAE.

Aim of Project

The main aim of the project is to design a novel Deep learning method at the Feature level. Variational Autoencoder is used in this project for generating test cases, these test cases need to have a ground truth label based on the r-separation distance. For each one we can query a ML model and its testing to see if the predicted label is the same as the ground truth label therefore, we will have our accuracy.

Research Objectives

Objective1: To learn the basics of the Deep Learning Robustness and Adversarial Examples.

Objective2: In this project we need a generative model to generate test cases for this we will use a generative model called Variational Autoencoder

Objective3: To decide the ground truth label which is the oracle of the generated test cases from ther-separation distance.

Objective4: Testing

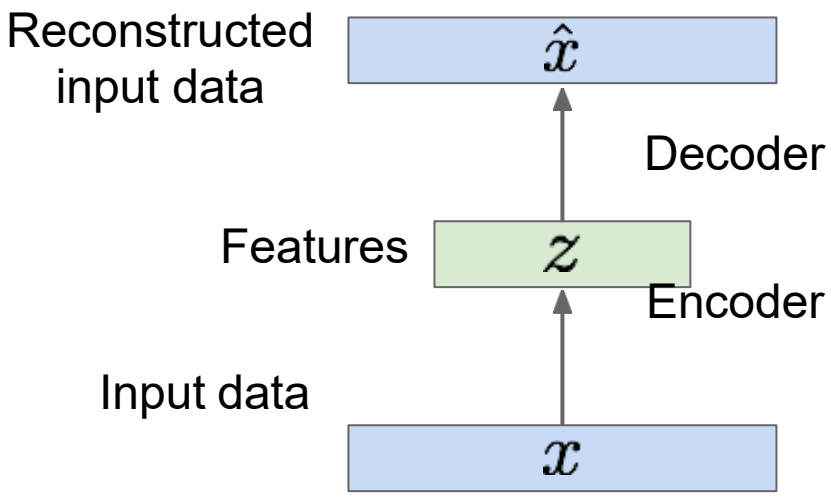
Key Literature and Background Reading

Various research papers have been studied in relation to Adversarial Examples, Deep Learning testing techniques and Generative models considering different approaches.

Study on Autoencoders *(AE) and Variational Autoencoder (VAE)*

AUTOENCODERS:

VAE are related to a type of unsupervised learning model called autoencoders. So, with Autoencoders we don't use it do generate data but it's an unsupervised approach for learning a lower dimensional feature representation from unlabelled training data. We have our input data x and we have some features we call z and then we will have an encoder that's going to be a function mapping from this input data to this feature z. This encoder can take many different forms they would generally use neural networks for this but originally these models have been around for a long time. In the 2000s it was used with linear layer and non-linearities and later it was used with fully connected deeper networks. Recently there was been an increasing use of CNNs for these encoders.



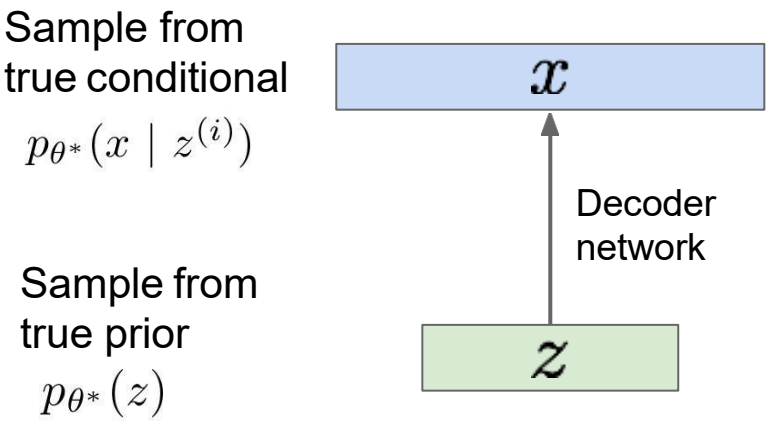
So, we take our input data X and we map this to some feature z. We usually specify this Z to be smaller than x and we perform basically dimensionality reduction because of that. Now the question is Why do we want z smaller than x here? it's because z should represent most important features in x., we want z to be able to learn features that can capture meaningful

factors of variations in the data. The way ae can learn feature representation is that we train the model such that the features can be used to reconstruct our original data. So we want to have input data that we use an encoder to map it to some lower dimensional features Z which is the output of the encoder network and we want to be able to take these features that were produced based on this input data and then use a decoder a second network and be able to output now something of the same size dimensionality as x, so we want to be able to reconstruct the original data. And again, for the decoder we are basically using same types of networks as encoders so it's usually a little bit symmetric and now we can use cnn networks for most of these. So, to summarise, our input data we pass it through our encoder first which is going to be something like a four-layer convolutional network and get these features and then we are going to pass it through a decoder which is a four layer for example an upconvolutional network and then get a reconstructed data out at the end of this. And the reason we have a convolutional network for the encoder and an upconvolutional network for the decoder is because at the encoder we are basically taking it from this high dimensional input to these lower dimensional features and now we want to go the other way go from our low dimensional features back out to our high dimensional reconstructed input. And so in order to get this effect that we wanted before of being able to reconstruct our input data we use a loss function e.g. say L2. An important thing to note here that even though we have a loss function here, there's no external labels that are being used in training this. we only have our training data that we are going to use it both to pass it through the network as well as to compute our loss function.

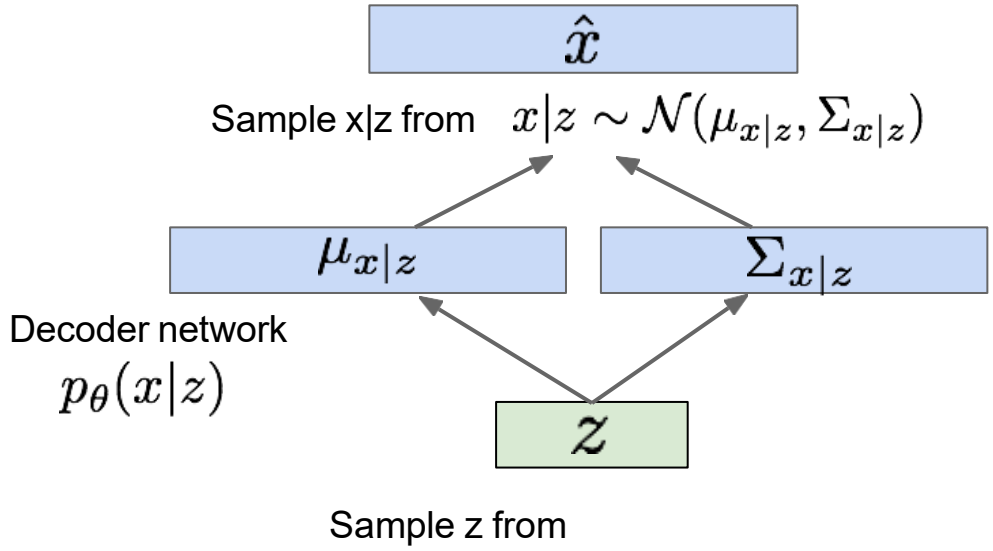
Variational Autoencoder

Variational autoencoders which is a probabilistic spin on autoencoders that will let us sample from the model in order to generate new data. So, in VAE, here we assume that our training data that we have $\{X^{(i)}\}^N$ is generated from some underlying, unobserved latent representation Z. Its this intuition that Z is some vector where each element of Z is capturing how little or how much of some factor of variation that we have in our training data. Now the generation process is sample a prior over Z. e.g. A gaussian is something that's a natural priorthat we can use for

factors of Z and then we are going to generate our data X by sampling froma conditional distribution p(x | Z). So, we have parameters of our prior and conditional distributions and in order to have a generative model be able to generate new data we want to estimate these parameters of our true parameters. Since we're modeling probabilistic generation of data, encoder and decoder networks are probabilistic.



So, to represent a model for this generator process we can choose our prior p(z) to be something simple e.g., Gaussian. Now for our conditional distribution p (x|z) , this is much more complex because we need to use this to generate an image and we can represent this with a neural network and this is called the decoder network i.e. we are taking some latent representation and trying to decode this into the image that its specifying. We would want to train this model so we can learn an estimate of these parameters. Once we have trained our VAE, we can just use the decoder network to generate our data. In Variational Autoencoders there are two main loss functions KL (Kullback Leibler) divergence and reconstruction loss. KL divergence shows dissimilarity between two different distributions. So, KL divergence between (P || Q) is the measure of dissimilarity between these two distributions. KL divergence is always >= 0. KL divergence is not symmetric Kl (P||Q) != KL(P||Q). We will try to minimize the sum of reconstruction loss and the KL divergence loss in the Variational Autoencoders.



Kingma and welling, “Auto-Encoding Variational Bayes”, ICLR 2014

Adversarial Examples

Adversarial examples are specified inputs created with the intention of fooling the neural network to misclassify the given input.
The epsilon-bounded example x' that meets the following criteria is a common definition of an adversarial example.

$$f_\theta(\boldsymbol{x'}) \neq y$$

That is, the classifier with parameters, theta, does not decide the actual label, y. A bounded adversarial scenario has the same characteristics.

$$\left\| \boldsymbol{x'} - \boldsymbol{x} \right\|_p < \epsilon$$

Ethical Use of Data

In this task, Deep learning algorithm is executed for testing image data at the feature level.
A generative type of model called Variational Autoencoder was trained. As for this project image datasets that used were MNIST, FashionMNIST and CIFAR10. The datasets were available in the pytorch module and also on Kaggle. As all of these datasets are Open datasets and are available freely on the public domain. All the Ethical guidelines were properly followed for this project.

Design

Due to recent advancements in the use of deep learning and machine learning techniques in many import aspects of life there is a need to test accuracy and robustness of deep learning techniques. For this project we use a type of generative model called variational autoencoder which will be implemented in python programming language and using pytorch module. The general architect of Variational Autoencoder includes and encoder and a decoder network which can be represented by neural networks. As we will be working with the Image datasets our encoder and decoder consists of convolutional layers. The model also consists some important functions such as reconstruction loss and KL divergence loss. The data is passed through the Encoder network and it is then compressed and it gives us mean and variance. By using mean and variance we can regenerate the data using the decoder network. We also find the r separation distance in the latent space which will later be used to regenerate the data.
Some import Functions in our VAE model:

Dataset Description and Data Pre-processing

For this research we are using a standard open dataset called MNIST. Dataset consists of handwritten digits of classes from 0 to 9. For this project we do not need to pre-process the data as we will be passing all the data through the Variational Autoencoder and then try to reconstruct the data.

Implementation

This project was implemented in the Python programming language using the pytorch module. Pytorch was selected because we can have access to GPU.

Python libraries used in this project are as follows:

1)Torch:

For variable, Normal distribution, KL diversion, datasets

2) Numpy:

Numpy is used for matrices, arrays and mathematical calculations.

3)Matplotlib:

Matplotlib for plotting losses, results and charts.

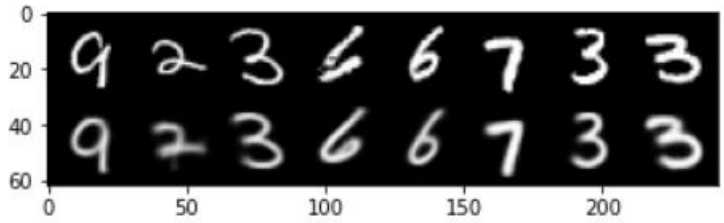
4) Pandas:

Creating and handling data-frames and data-structures.

5)TQDM:

To create progress bars for monitoring progress.

First, we start by choosing the dataset for this project. As this was related to image generation, Standard datasets such as MNIST, FashionMNIST, CIFAR10 were selected. As for the generative model Variational autoencoder was choses due its latent space properties and the decoder network which is used as a Generator. We define a Class VAE and define components of VAE as functions. Data is then passed through the encoder network and it is trained for 50 epochs. R separation distance is measured in the latent feature space of VAE and this distance is used to generate images. Some random samples were generated from the trained model.



Now as the model is trained, we try to manipulate from the latent space. We select two inputs in the I-infinity norm and then try to find the distance between these points in the latent space. If the distance is less than the 2r then we would generate the image else would discard it. Then repeat this step for all the points in the dataset. We were able to generate images that were within this r distance and store it in the results folder in the directory.

Some of the important functions in our algorithm.

Self.encoder:

This is our encoder function which is represented by a convolutional neural network. ReLU activation function is used to activate the layers. This function takes input data and compresses it.

```
self.encoder = nn.Sequential(
    nn.Conv2d(input_dim, dim, 4, 2, 1),
    nn.BatchNorm2d(dim),
    nn.ReLU(True),
    nn.Conv2d(dim, dim, 4, 2, 1),
    nn.BatchNorm2d(dim),
    nn.ReLU(True),
    nn.Conv2d(dim, dim, 5, 1, 0),
    nn.BatchNorm2d(dim),
    nn.ReLU(True),
    nn.Conv2d(dim, z_dim * 2, 3, 1, 0),
    nn.BatchNorm2d(z_dim * 2)
)
```

Self.decoder:

This is our decoder function which is also represented by a convolutional neural network.

```
    self.decoder = nn.Sequential(
        nn.ConvTranspose2d(z_dim, dim, 3, 1, 0),
        nn.BatchNorm2d(dim),
        nn.ReLU(True),
        nn.ConvTranspose2d(dim, dim, 5, 1, 0),
        nn.BatchNorm2d(dim),
        nn.ReLU(True),
        nn.ConvTranspose2d(dim, dim, 4, 2, 1),
        nn.BatchNorm2d(dim),
        nn.ReLU(True),
        nn.ConvTranspose2d(dim, input_dim, 4, 2, 1),
        nn.Tanh()
    )
```

ReLU and tanh functions are used activation functions. By the help of this function, we will be able to reconstruct our data.

def encode:

This function takes our input data x and after encoding return the mean and variance. We take the log of variance to make it a positive value.

```
def encode(self, x):
     # encode x
    mean, logvar = self.encoder(x).chunk(2, dim=1)
    return mean, logvar
```

def decode:

This function samples from our latent space z and returns samples.

```
    def decode(self, z):
#         z = torch.reshape(z,(len(z), self.z_dim, 1, 1))
        samples = self.decoder(z)
        return samples
```

def sample:

Samples from our latent space and returns the corresponding image space map.

```
  def sample(self, num_samples, cuda):
      """
      Samples from the latent space and return the corresponding
      image space map.
      :param num_samples: (Int) Number of samples
      :param current_device: (Int) Device to run the model
      :return: (Tensor)
      """
      z = torch.randn(num_samples,
                      self.z_dim,2,2)

      if cuda:
          z = z.to('cuda')


      samples = self.decoder(z)
      return torch.flatten(z, start_dim=1), samples
```

def generate:

We add some perturbation and given an input image x we return the new generated image.

```
def generate(self, x):
    """
    Given an input image x, returns the new image
    :param x: (Tensor) [B x C x H x W]
    :return: (Tensor) [B x C x H x W]
    """
    # encode x
    mean, logvar = self.encoder(x).chunk(2, dim=1)

    q_z_x = Normal(mean, logvar.mul(.5).exp())
    q_per = Normal(torch.zeros_like(mean),0.01*torch.ones_like(logvar))

    # add perturbation
    z_new = q_z_x.rsample() + q_per.rsample()

    # reconstruct x from z
    x_new = self.decoder(z_new)

    return x_new
```

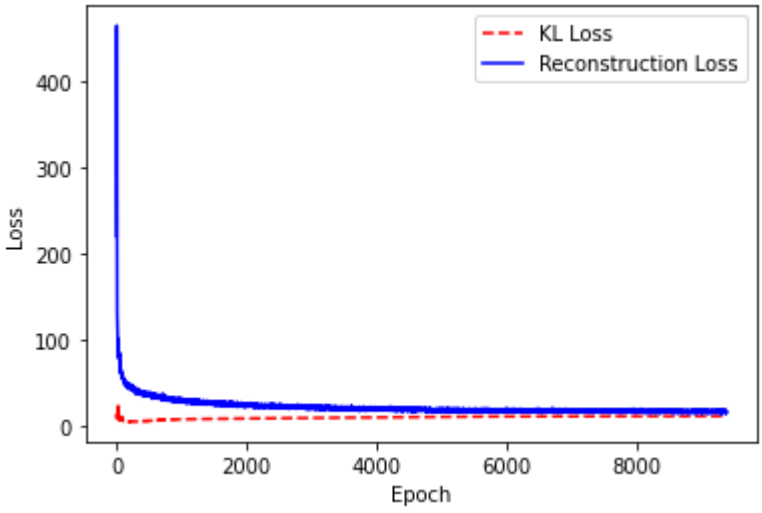
def save_generated_img:

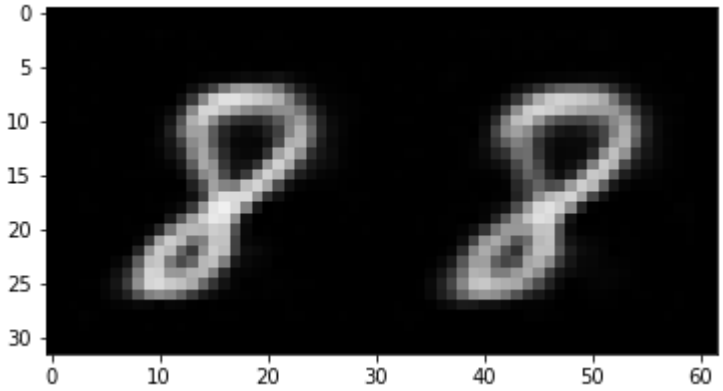
This function creates a folder in the root directory and stores all the generated images.

The main aim of the project was to learn the basics of Deep learning robustness and Adversarial examples. I had to research and learn about the basics of adversarial examples and generative model. Variational autoencoder was selected because of its generation properties and its latent space.

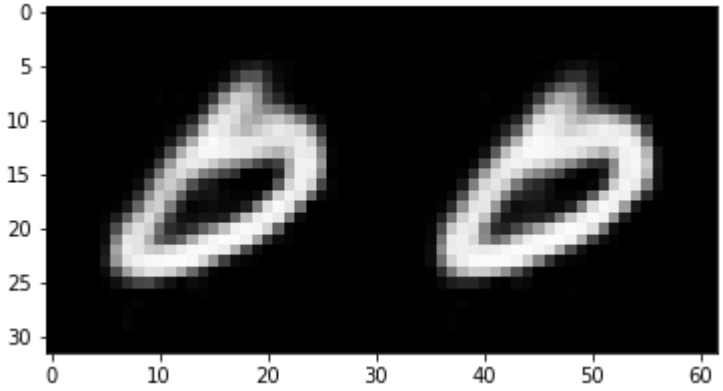
Evaluation

In this section we compare the image generated within the r distance with the input image and we can see that both the images share the same ground truth label. Also we can see the graphs plotted of KL divergence loss and the reconstruction loss.





Input Image(left), Reconstructed image within the r distance(Right)



Learning Points

Through this research project several theoretical concepts and practical implementation were learned.

- Collection of authenticated datasets by following ethical considerations for these types of projects.
- Different data compression and data reconstruction techniques.
- Practical implementation of plotting graphs and results using python libraries.
- Learned about different types of Generative models.
- Importance of Deep learning testing.
- Learned the r separation property.
- How deep learning techniques are prone to adversarial attacks.
- Learned about finding distances in n dimensions.
- Learned about the importance of dimensionality reductions
- Learned about the adversarial examples.
- Initialization, implementation of data fitting in deep learning algorithms and how to evaluate them is learned from this task.
- Learned about graphical and probabilistic approaches.
- Noise for generating images.
- Hyperparameter tuning of the algorithms.
- Time management and Importance of schedule and to always have a plan B.

Conclusion and Future Scope

Due to developments in advanced deep learning and computer vision applications, it is essential to learn the basics of robustness and accuracy of these systems. The main objectives of this research were to study a generative model and generate test cases through the r separation property. For this, I was able to learn the Variational autoencoder and measure the r separation distance in the latent space of VAE and generate an image within the r distance of the seed input. The generated images were more robust. The r separation property is very useful and it helps us in deciding the ground truth. The generated images also shared the same ground truth label as the input seed. Further Research can be done in this direction with high computing power and by studying more about adversarial examples and different types of generative models.

Bibliography

[1] Diederik P Kingma, Max Welling, Auto-Encoding Variational Bayes, *20 Dec 2013*, eprint arXiv:1312.6114

[2] Yao Li, Tongyi Tang, Cho-Jui Hsieh, Thomas C. M. Lee, Detecting Adversarial Examples with Bayesian Neural Network, 28 May 2021, arXiv

[3] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z. Berkay Celik , Ananthram Swami , Practical Black-Box Attacks against Machine Learning, 19 March 2017, arXiv

[4] Ian J. Goodfellow, Jonathon Shlens & Christian Szegedy, EXPLAINING AND HARNESSING ADVERSARIAL EXAMPLES, 20 March 2015, arXiv

[5] S. Dola, M. B. Dwyer and M. L. Soffa, "Distribution-Aware Testing of Neural Networks Using Generative Models," 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE), 2021, pp. 226-237, doi: 10.1109/ICSE43902.2021.00032.

[6] Lawrence Cayton. 2005. Algorithms for manifold learning. Univ. of California at San Diego Tech. Rep 12, 1-17 (2005).

[7] Carl Doersch. 2016. Tutorial on Variational Autoencoders. ArXiv (2016). arXiv:1606.05908

[8] Ian Goodfellow et al., “Generative Adversarial Nets”, NIPS 2014

[9] Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014

[10] Radford et al, ICLR 2016

[11 WEI HUANG , XINGYU ZHAO , ALEC BANKS , VICTORIA COX , XIAOWEI HUANG “Hierarchical Distribution-Aware Testing of Deep Learning”

[12] Zhengli Zhao, Dheeru Dua, Sameer Singh, ” GENERATING NATURAL ADVERSARIAL EXAMPLES”

[13] T. Byun and S. Rayadurgam, "Manifold for Machine Learning Assurance," 2020 IEEE/ACM 42nd International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER), 2020, pp. 97-100.

[14] Yao-Yuan Yang, Cyrus Rashtchian, Hongyang Zhang, Ruslan Salakhutdinov and Kamalika Chaudhuri, “A Closer Look at Accuracy vs. Robustness”