Vapnik-Chervonenkis Dimension in Neural Networks

by

Weiting Liu
Lakehead University

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTERS

in the Department of Computer Science

Lakehead University

Vapnik-Chervonenkis Dimension in Neural Networks

by

Weiting Liu
Lakehead University

Supervisory Committee

Dr. Yimin Yang, Supervisor
(Department of Computer Science, Lakehead University, Canada)

Dr. Ruizhong Wei, Co-Supervisor
(Department of Computer Science, Lakehead University, Canada)

ABSTRACT

This thesis aims to explore the potential of statistical concepts, specifically the Vapnik-Chervonenkis Dimension (VCD)[33], in optimizing neural networks. With the increasing use of neural networks in replacing human labor, ensuring the safety and reliability of these systems is a critical concern. The thesis delves into the question of how to test the safety of neural networks and optimize them through accessible statistical concepts.

The thesis presents two case studies to demonstrate the effectiveness of using VCD in optimizing neural networks. The first case study focuses on optimizing the autoencoder, a neural network with both encoding and decoding functions, through the calculation of the VCD. The conclusion suggests that optimizing the activation function can improve the accuracy of the autoencoder at the mathematical level.

The second case study explores the optimization of the VGG16 neural network by comparing it to VGG19 in terms of their ability to process high-density data. By adding three hidden layers, VGG19 outperforms VGG16 in learning ability, suggesting that adjusting the number of neural network layers can be an effective way to analyze the capacity of neural networks.

Overall, this thesis proposes that statistical concepts such as VCD can provide a promising avenue for analyzing neural networks, thus contributing to the development of more reliable and efficient machine learning systems. The final vision is to allocate the mathematical model reasonably to machine learning and establish an idealized neural network establishment, allowing for safe and effective use of neural networks in various industries.

Contents

List of Tables

List of Figures

ACKNOWLEDGEMENTS

I would like to thank:

Dr. Yimin Yang, my supuervisor,I am super thankful for his invaluable guidance and support. Being an undergraduate student who is majoring in information and computing science, I have to say it is a hard time when I first started my first seminar. Dr. Yang did not let that become a barrier to my education and provided his students all with a pure study enviroment, education, and a strong group learning atmosphere during my time at Lakehead University. As a result of this experience, I have gained a different perspective across Canada. I am thankful for Dr. Yang's patience and tolerance, which have contributed significantly to my academic success.

DEDICATION

"What we find changes who we become."

- Peter Morville

I would like to dedicate this thesis to,

Firstly, I express my heartfelt gratitude to my family who stood by me during the unprecedented Covid-19 pandemic. Their support helped me to be grateful to face the challenging times and continue with my academic tour. Their belief gave me the biggest support for finish this through this hard time.I am grateful for their unconditional love and sacrifices, which have made this achievement possible.

Secondly, I extend my appreciation to Vetor Institute Canada for the Vector Scholarship in AI that support my study.

In particular, I would like to say thanks again to my supervisor Dr. Yimin Yang , and co-supervisor Dr. Ruizhong Wei for their invaluable guidance and support throughout the course of this research. I admire their research on machine learning and their rigorous attitude in class. And in their courses, students can really learn knowledge.

In conclusion, this thesis is a culmination of the support, encouragement, and guidance of various individuals, and I am forever grateful to all of them.

Chapter 1

Introduction

1.1 Overview

For anyone in today's society, the words machine learning(ML) and artificial intelligence(AI) will not be unfamiliar. Especially in recent years, the rise of new technologies such as unmanned driving and medical judgment has reminded people to realize this face again that technology has changed our lives. However, how do things change so fast, and actually, ML and AI just appear not longer than 100 years.

Firstly, researchers focused on "reasoning," which means how the machine makes logical decisions based on a set of rules. After that, researchers turned their attention to "knowledge," which means how can machines store and access vast amounts of data. In recent years, the focus has shifted again to "learning," means how can machines automatically improve performance over time.[14][15]

Machine learning(ML) and neural network have now become a complex field with cutting-edge and unimaginable techniques that can be applied in different fields. The essence of machine learning is interdisciplinary and uses knowledge beyond probability theory, statistics, approximation theory, and convex analysis. Its main goal is to achieve the best results by designing algorithms and adopting different techniques on different problems. Machine learning algorithms are designed to analyze data, and patterns are once again the focus of machine learning. Promising machine learning algorithms use it to make predictions on new data.[5]

Statistical theories play a critical role in machine learning, and the field is closely related to inferential statistics, or statistical learning theory. Machine learning theory places particular emphasis on the development of achievable and effective learning algorithms that can prevent error accumulation. For part of the reasoning process, machine learning cannot replace the human brain for the time being. Therefore, a huge part of the current study is basically focused on the approximate algorithms, which can provide available solutions to complex problems.[3]

A significant portion of machine learning technology can be considered well-established, including data mining, computer vision, natural language processing, biometric recognition, search engines, medical diagnosis, credit card fraud detection, securities market analysis, DNA sequencing, speech and handwriting recognition, games, and robots. Due to its versatility and wide applicability of it, our lives are all around with it. Even we need to constantly update technology to avoid the discomfort caused.[16]

In the field of data mining, ML algorithms are used to extract useful information from plenty of datasets which is also complex. By analyzing patterns and relationships, the model can be used to identify trends, predict future outcomes, and make data-driven decisions. In computer vision, ML is used to analyze images and videos, recognize objects and faces, and extract useful information from visual data.[17]

Natural language processing (NLP) is another area where autoencoder, one of ML models, has been largely utilized. NLP involves allowing machines to understand human language and generate human language using a human voice. This is not just as simple as inputting language text and generating it, but more focus is on generation and understanding. It is similar to the part of the human brain when newborns need to learn and practice languages. ML algorithms can be used to analyze large volumes of text and generate meaningful insights.

NLP is also the fastest-growing part of biometric recognition. Biometric recogni-

tion is an area where ML has been widely used in recent years. Phones can accurately unlock by using physical characteristics such as fingerprints and facial features. Not only for phones but also has major implications for security, law enforcement, healthcare and finance, among other things.[18]

In finance, ML is used to analyze market data and make predictions about future market trends. Before the global financial crisis comes, or before major investment decisions, by identifying patterns and relationships in financial data, ML can avoid risks through scientific means.

In the field of robotics, ML can achieve varying degrees of learning effects by writing programs and practicing them. It includes navigating, recognizing objects and faces, and interacting with humans in a human way. By combining machine learning with other technology such as computer vision and natural language processing, robots can be designed to perform a variety of tasks in many different environments.

1.1.1 Machine Learning in Medical Decision-Making

In recent years, ML also being a powerful tool in medical research and healthcare, with applications ranging from image analysis and diagnosis to drug discovery and personalized medicine.

In radiation oncology, ML has the potential to transform the practice of radiation therapy from treatment planning to outcome prediction. Radiation therapy is a common form in that more than half of cancer patients receive ionizing radiation as part of their treatment. Radiation therapy involves a multitude of processes, not only from consultation to treatment but beyond to ensure that patients receive the prescribed dose of radiation and respond well. These processes vary in complexity and may involve several stages of complex human-machine interaction and decision-making, which will naturally require the use of ML algorithms to optimize and automate these processes.[19]

Based on these trends and advancements, it is plausible to predict the emergence of several new technologies, as examples shown in the figure. Additionally, it is noted that biologically-inspired AI will play a significant role in the development of ML models in the future.

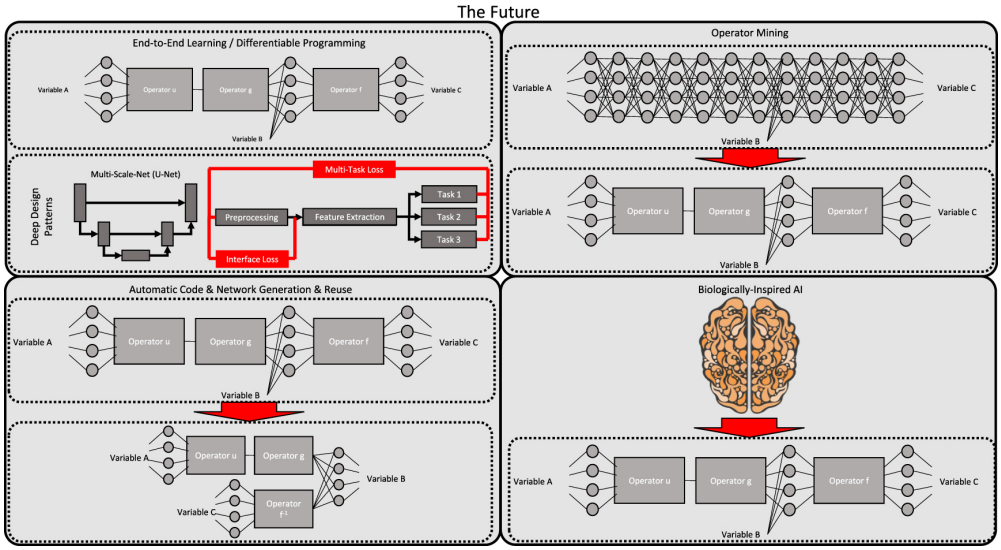

Figure 1.1: **New technologies in medical area based on machine learning[19]**

1.1.2 The Intersection of Geology and Machine Learning

In Karimpouli's 2020 article[20], he first discusses the practicalities of using ML techniques in geology. Its application has the potential to restructure the way researchers analyze data, helping them understand the properties of different complex geological materials differently.

In this field, the first main challenge need to be mensioned is the geological characterization. For instance, when researchers want to analyze the properties for a specific rock formation, there may be some obvious variations in the composition of the rock at different points within the formation. And it may be troubling because the limitations of data and the nature of the measurements themselves. But ML can help address this challenge by providing more accurate and reliable predictions of geological properties based on limited data sets.

In particular, Karimpouli noted that ML can be used to quantify uncertainty in complex geological materials and to model fractures and phase transitions in these materials. Its ability to learn from previous data and generalize to new environments is one of its advantages. This is particularly useful when there is little data or the data is highly variable. By learning from previous data, ML can identify patterns and relationships that humans might not immediately spot.

Another advantage is the ability to automatically analyze large volumes of data. Nowadays, the traditional methods of geological analysis are time- waste and labor-

intensive. It may require researchers to have significant expertise. ML algorithms can automate these and allow researchers to have more time to efficiently analyze large volumes of data. Whatsmore, they can also identify patterns and trends that traditional methods might struggle to discern.

Despite its many positive side, there are still some difficulties that must be addressed. One of these is for now with ML, it needs high-quality data. This is also an especially challenge in geology, where data may be limited or difficult to obtain. Additionally, there is a need for ML models that are robust and can account for the complexity of geological materials.

To address some of these issues, Karimpouli proposed the use of Gaussian processes and physical informatics ML as a smart meshless approach to solving the seismic wave equation. The proposed architecture of a physically informative ML model for solving the 1D seismic wave equation is shown in the figure. By exploiting its powerful capabilities, Karimpouli's method provides a promising framework for developing more accurate and reliable geological ML models.

Overall, the use of ML based on theoretical knowledge of physics has the potential to revolutionize geology. It can help solve the challenges associated with the uncertainty and variability of geological data and automate the analysis of large volumes of data. While there are still challenges that must be addressed, the proposed framework for machine learning using Gaussian processes and physio informatics provides a promising avenue for developing more accurate and reliable geological ML models.[20]

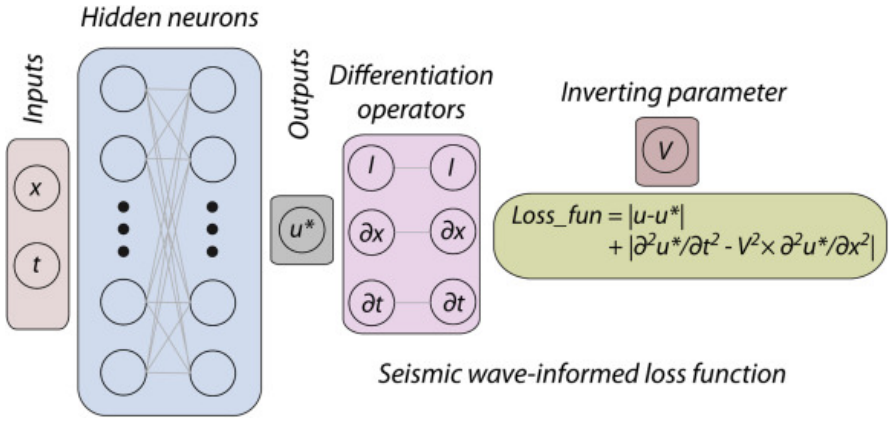

Figure 1.2: **Architecture of the proposed PIML for solving 1D seismic wave equation.**[57]

Various experiments shown in this article were conducted to analyze the effect of variables, such as the number of neurons on the prediction accuracy of noise-free

data using this final model. The comparison of absolute error is shown in Figure 2, which indicates the potential of physics-based deep learning methods for more accurate predictions.

Also, the physical laws and empirical relations are incorporated into the learning process. These methods show better performance when having more analytically tractable, which allows researchers taking a good understanding of the underlying physical processes.

To further demonstrate the potential of physics-informed machine learning methods in geosciences, the authors evaluated their power and flexibility in solving seismic wave equations. The results showed that these methods can provide accurate and efficient solutions, demonstrating their effectiveness in solving complex geoscientific problems. This makes the potential of these methods for further applications highlighted in other fields of geoscience research.

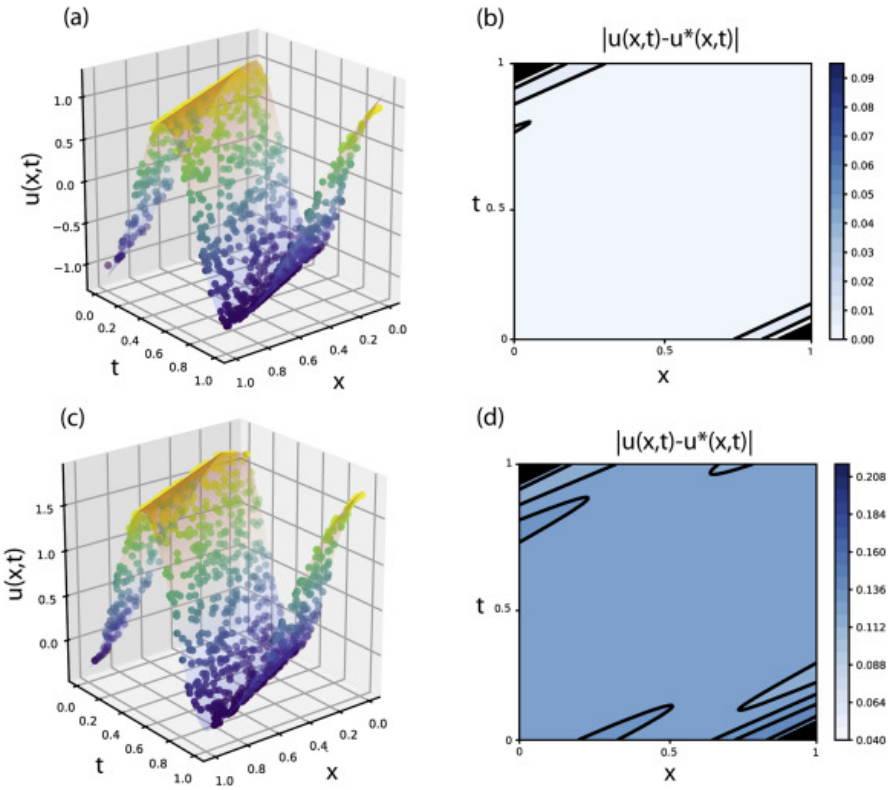

Figure 1.3: **The PINN evaluation: (a) Predicted $u(x,t)$ and (b) the absolute error of predictions for noise free data. (c) Predicted $u(x,t)$ and (d) the absolute error of predictions for noisy data. $u(x,t)$ is the output, $u^*(x,t)$ is the output of network.[57]**

ML grows rapidly with countless applications in various industries, including healthcare, finance, and transportation. At its core, it involves training algorithms to learn from data and make predictions or decisions without being explicitly programmed. One popular type of this is the neural network.

A neural network(NN) is a computational model that imitates the biological neural network, which is the central nervous system of animals, especially the structure and function of the human brain. NN is computed by a large number of artificial neuron connections, and they have the function of learning. In other words, they can change their internal structure on the basis of external information, and they are adaptive systems.

The modern NN is a nonlinear statistical data modeling tool. NN is optimized through a learning method based on mathematical statistics, which makes them a practical application of mathematical statistics. By using this method, we can obtain a large number of local structure spaces that can be expressed by functions.

In the field of artificial perception in artificial intelligence, we can make decisions about artificial perception through the application of mathematical statistics. Through statistical learning, the artificial neural network can have simple decision-making ability and simple judgment ability similar to human beings. This method has more advantages than formal logical reasoning and calculation. Like other ML methods, NN has been used to solve a variety of problems, such as machine vision and speech recognition. These problems will be more difficult when we used traditional rule-based programming.

In recent years, NN's potential has been explored mostly in the healthcare field. The use of neural networks in this area mainly has been used for revolutionizing the way that doctors diagnose and treat diseases. It shows that ML gives promising potential in improving various aspects of radiation oncology, particularly in the optimization and automation of processes, such as treatment planning and outcome prediction. Radiotherapy, as one of the most common treatment modalities for cancer patients, involves a complex set of processes that require sophisticated human-machine interactions and decision-making. With ML algorithms, these can be optimized and automated. This means it will make sure that patients receive the appropriate radiation dose and respond well to the treatment. One study published in the Journal of Cancer explored the use of genotype information and a mathematical model to precisely predict radiation pneumonitis, a common side effect of lung cancer treatment. The results showed that the proposed model gives better performance than traditional

models in predicting radiation pneumonitis, and demonstrated the potential of ML in improving the efficacy and safety of radiotherapy for lung cancer patients.[21]

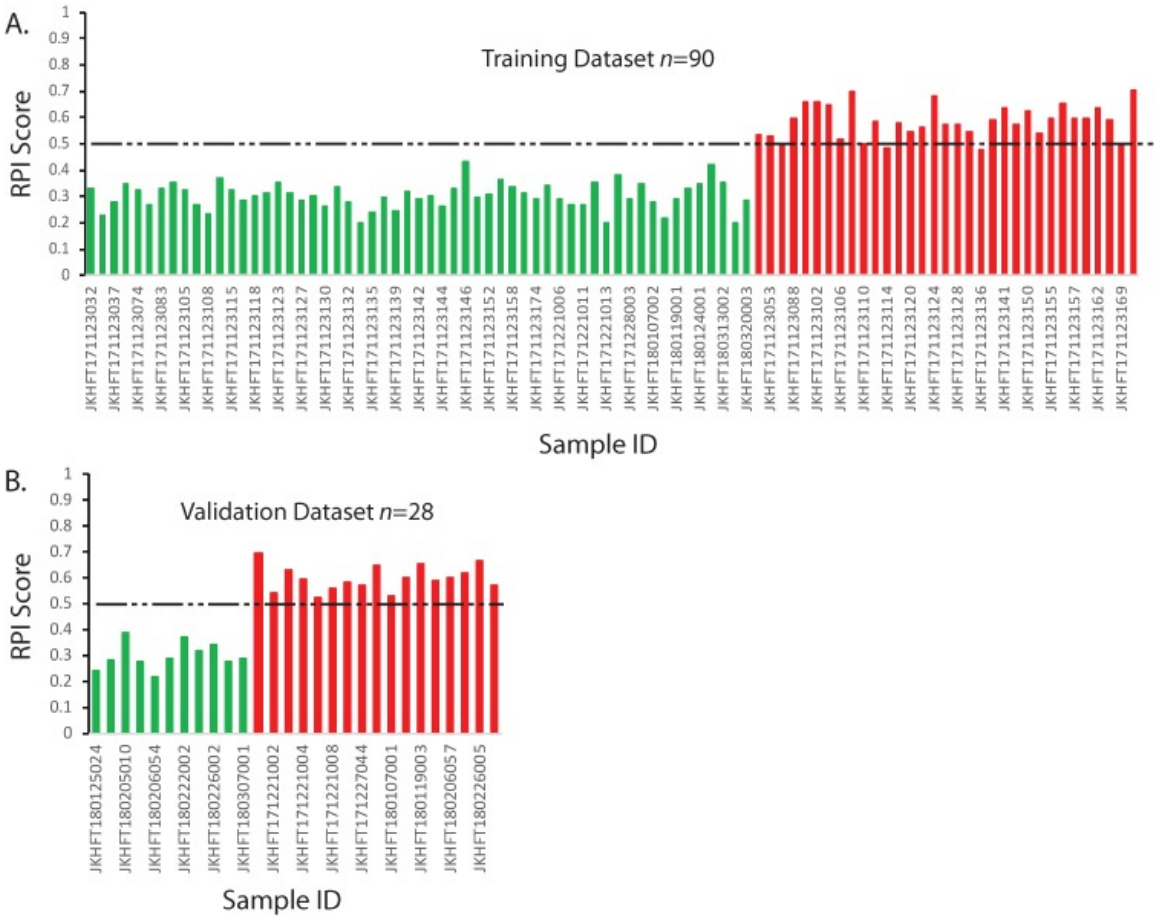

Figure 1.4: **Prediction model from Du[21]**

In another study by Yoon[22], the researchers used a deep learning model to predict tumor response to radiotherapy in patients with head and neck squamous cell carcinoma (HNSCC). The researchers used a dataset of 185 patients with HNSCC who received radiotherapy and developed a neural network model to predict tumor response.

After the training and testing, researchers found that the model can achieve accuracy up to 0.74, whats more, the area under the receiver operating characteristic curve can achieve to 0.77. These results show that this model has the potential to accurately predict tumor response to radiotherapy and could be a valuable tool for optimizing treatment plans.

1.2 Motivation

Recent advancements in the field of ML have demonstrated its indelible role of itself and big data in the success of neural network applications. However, the era of big data has arrived, and it is clear that ML needs vast amounts of data for training and optimization. While some data collections can be obtained without incurring significant costs of time and money, such as the widely-used iris database, other data sets require extensive efforts and resources to obtain, such as car accident pictures, road surface recognition images for unmanned driving systems, and seismic wave data. The availability and quality of data can significantly affect the accuracy of a neural network, as the authenticity and error values of input data can become uncertain factors that restrict the performance of the network.

This thesis aims to analyze and optimize neural networks including autoencoders and VGG networks using statistical concepts. The neural network will be optimized by calculating and determining the known database to achieve higher accuracy. Autoencoders can be used for more areas after analysis and optimize. And for the VGG network, VCD can use to analyze its structure and create another neural network that can have the same performance or even better. Under these two purposes, selecting a more suitable activation function can further increase the accuracy of the network.

While large-scale neural network models have been extensively employed to achieve high generalization performance, it does not necessarily reflect the quantifiable, interpretable, and mathematical capacity of a network as the parameters of the networks increase. Given the increasing demands of asking for more explainable AI models, the thesis aims to find a quantifiable capacity of neural network models under different network architectures and the size of free parameters using VC dimensions. Therefore, the results of the thesis could be used to compare the expected performance with different network models.

1.3 Problem Description

The research problem addressed in this thesis is the integration of statistical concepts into neural networks and their application. The focus is on understanding and utilizing the VCD in statistics, which has been identified as relevant to the optimization of specific neural networks. The objective is to analyze a single neural network through the application of VCD and statistical concepts and eventually scale it up to a mature

neural network.

The VCD is a measure of the representational ability and complexity of a learnable classification function space in VC theory. It is determined by the maximum number of points that a given algorithm can classify correctly. The power of a classification model is directly related to its complexity. For example, a high-degree polynomial classification model can accurately fit a set of points. However, this same model may also misclassify other point sets that follow a different pattern. Therefore, such a model is considered highly capable but may not be the most optimal solution.

By integrating statistical concepts into neural networks and considering VCD, this research aims to optimize neural networks and improve their accuracy and effectiveness. The successful application of these concepts will ultimately reduce the dependence on human expertise in fields such as medical diagnosis and early prediction of natural disasters.

1.4 Contribution

This thesis is structured into two main parts. The first part aims to examine the optimization capabilities of the VCD in known deep-learning models. Idealized digital deductions will be utilized to obtain implementable optimization results. Firstly, the analyze of a single hidden layer neural network using VCD will be expanded to multi-layer hidden layers. Then, the unique structure of the autoencoder will be leveraged to optimize VCD for the autoencoder.

The second part of the study will focus on the mathematical deduction of the sub-parts of the VGG network. The mathematical proof will be used to demonstrate the varying effects of the VGG16 and VGG19 network structures. Subsequently, theoretical optimization of the VGG series network will be deduced from the mathematical proof. Overall, this research aims to explore the potential of VCD as an optimizer for deep-learning models and to provide theoretical insight into the optimization of specific neural networks.

1.5 Organization of Thesis

This section was all about the introduction and the rest of the thesis proceeds as follows,

Chapter II, explained the background and relevant information of this study. It

mainly introduces the application of VCD in statistics and computers. After the introduction, complete the preliminary understanding of VCD.

Chapter III, introduces how VCD will be used as an analyzer in an autoencoder. This chapter presents a detailed account of the statistical definition of VCD in the background, and elaborates on the effective application of VCD to a single-layer neural network. Moreover, section 3.3 demonstrates the application of VCD principles in optimizing an autoencoder. It mainly includes how to choose the activation function and other optimization methods and the final effect display.

Chapter IV, it is further introduced that when VCD is applied by a more novel neural network, whether the optimization effect similar to the automatic encoder optimization can be obtained.This section is to elucidate the diverse models required for VCD implementation in different neural networks, as outlined in the background. Additionally, sections 4.3 and 4.4 explicate the distinct yet analogous application of VCD and offer optimization recommendations in two separate neural networks, which is VGG16 and VGG19 seperately. And whether the same theoretical support can be provided, and on the premise of theoretical support, how to optimize the new neural network with a higher utilization rate.

Chapter V explain the method and steps when using VCD to optimize any neural network in the future. Explain the problems that may arise during the process and the resulting results.

Chapter 2

Background and Related Work

2.1 Background

2.1.1 Statistical

Vapnik-Chervonenkis Dimension, in statistical learning theory, the VC dimension(VCD) is a measure of the capacity of a set of functions to fit a wide range of input patterns. It is defined as the largest number of points that can be shattered by the set of functions, meaning that the set can produce all possible dichotomies on those points.

The VCD plays a key role in understanding the connection between model complexity and generalization performance. Specifically, a set of functions with a higher

VCD can fit a wide range of functions but may cause overfitting in some situations. When a set of functions with a lower VCD may have limited the capacity to fit complex functions,and may also generalize better to new data.[2][3]

In statistical learning theory, VCD is often used to derive bounds on the generalization error of a learning algorithm, which provides a theoretical guarantee of its performance on unseen data. These are always based on the size of dataset, the complexity of model, and the VCD of the set of functions.[1]

In the context of statistical applications, the use of VCD involves a significant amount of mathematical terminology, which is an important consideration when defining VCD. It is essential to take note of these key points when citing the definition of VCD.

Before introducing VCD, it is essential to first understand a key definition: hypothesis space.

In the context of statistical learning theory, the hypothesis space in VCD refers to the set of all possible functions that can be used to model a dataset. It is a fundamental concept when analysis the complexity and capacity of machine learning models.

The specific definition of the hypothesis space can vary depending on the machine learning algorithm being used and the nature of the problem being addressed. In general, the hypothesis space includes a family of functions that can be parameterized and trained on a training dataset to optimize a specific objective function, such as minimizing the empirical risk or maximizing the likelihood of the data.[2]

The choice of hypothesis space is a critical consideration, as it determines the model's ability to approximate complex functions and generalize to new data. The VCD provides a quantitative measure of the capacity of the hypothesis space and can be used to analyze the trade-off between model complexity and generalization performance.

The next definition that needs to be mentioned in VCD is the shatter point.

In VCD, the shatter point refers to the largest number of data points that a hypothesis space can "shatter" or separate into all possible binary classifications. This means that the hypothesis space can produce all possible dichotomies (i.e., partitions of the data into two sets) on those points.

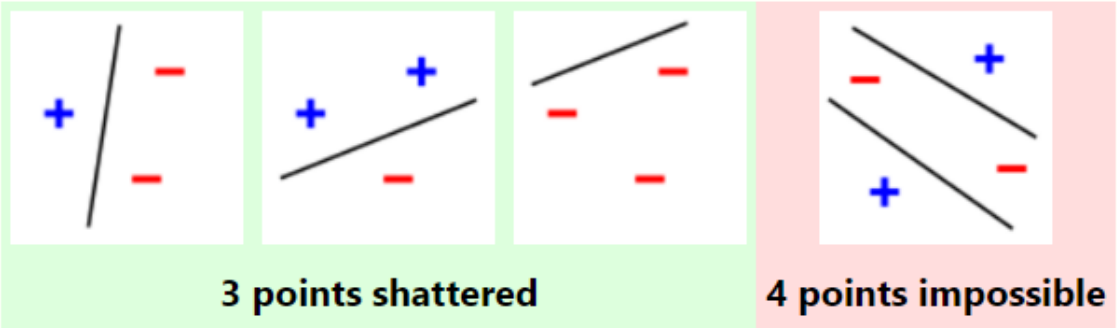

Figure 2.1: **Examples of points to be shattered[52]**

For example, if the shatter point of a hypothesis space is 3, it means that the space can perfectly separate any three data points into two sets (e.g., positive and negative). However, for four data points, there may be at least one set of points that cannot be separated by the hypothesis space. In other words, the hypothesis space is not powerful enough to correctly classify all possible combinations of four data points.

The shatter point is closely related to the VC dimension of a hypothesis space, as it provides an upper bound on the VC dimension. The VC dimension is the largest number of data points that a hypothesis space can shatter for any configuration of points, while the shatter point is the largest number of points that can be shattered for a specific configuration.

Understanding the shatter point is important in the analysis of the capacity and complexity of hypothesis spaces, as it provides a measure of their ability to fit complex functions and generalize to new data.[4]

How can VC dimension and related statistical definitions be applied in the process of machine learning? Is there a sound mathematical derivation system that connects neural networks with the statistical concept of VC dimension?

2.1.2 Netural Network

The neural network is a class of machine learning algorithms that imitate the structure and function of the human brain. It can learn complex patterns from data. Nowadays, NN has become a popular tool from image recognition and natural language processing to finance and healthcare.

NN is made with neurons that process and transmit information. Basically, each neuron in the neural network get input, applies an activation function, then produces

output that is transmitted to other neurons. The connections between each neuron are typically weighted, meaning that some inputs have a greater influence on the output than others.

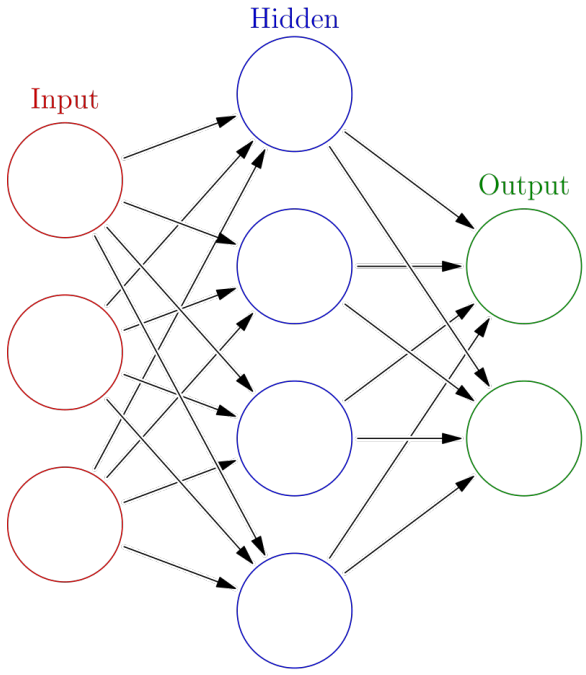

Figure 2.2: **Example of Netural Network**

One of the key advantages of NN is their learning ability from data. They can be trained on a dataset with labeled examples, and adjust their weights and parameters to minimize the difference between their predicted outputs and the true outputs. This process named backpropagation, involves propagating errors backwards through the network and adjusting the weights to minimize the error.

There are some typical structures of neural networks. Multilayer perceptrons (MLPs), are the most common and always used for classification and regression. Convolutional neural networks (CNNs) are specialized for processing images and other grid-like data. It is popular with image recognition and object detection. Recurrent neural networks (RNNs) are designed for processing sequences of data, such as text or time series, and have been used for such as language modeling and speech recognition area.

While neural networks have shown impressive performance, they also have limitations. One of the main challenges is overfitting, where the network becomes too complex and starts to memorize the training data rather than learning general pat-

terns. Regularization techniques, such as dropout and weight decay, can help to prevent overfitting.

Another challenge is interpretability, as neural networks are often considered to be black boxes, which makes trouble when humans want to understand how they can achieve the predictions. Recently, researchers are focused on developing methods for interpreting and visualizing neural networks, such as layer-wise relevance propagation (LRP) and saliency maps.

Now, neural networks are inside many areas of human life and have advances that can not be ignored in a wide range of applications. However, using them requires careful consideration, as well as strengths and limitations.[5]

Autoencoder

Among all neural networks, autoencoder is one of those that cannot be ignored. Its origins date can be found in 1980s, when it was first demonstrated that could learn representations of compressed data. Since then, autoencoders have gone through some developments, like some autoencoders based on different variations. It also become the preferred neural network in many fields.

The main idea of an autoencoder is to reconstruct an input from a low-dimensional encoding by training. The simple autoencoder shown in the figure can understand its two parts, an encoder that maps the input data to a lower dimension, and a decoder that maps the output of encoder can back to its original input dimension. In these two processes, the activation function used is also changed. The ultimate goal is that no matter the dimension reduce, its output data has the same dimension as the input data.[6] During training, the encoder and decoder are trained together by using backpropagation to minimize the difference between input and output.

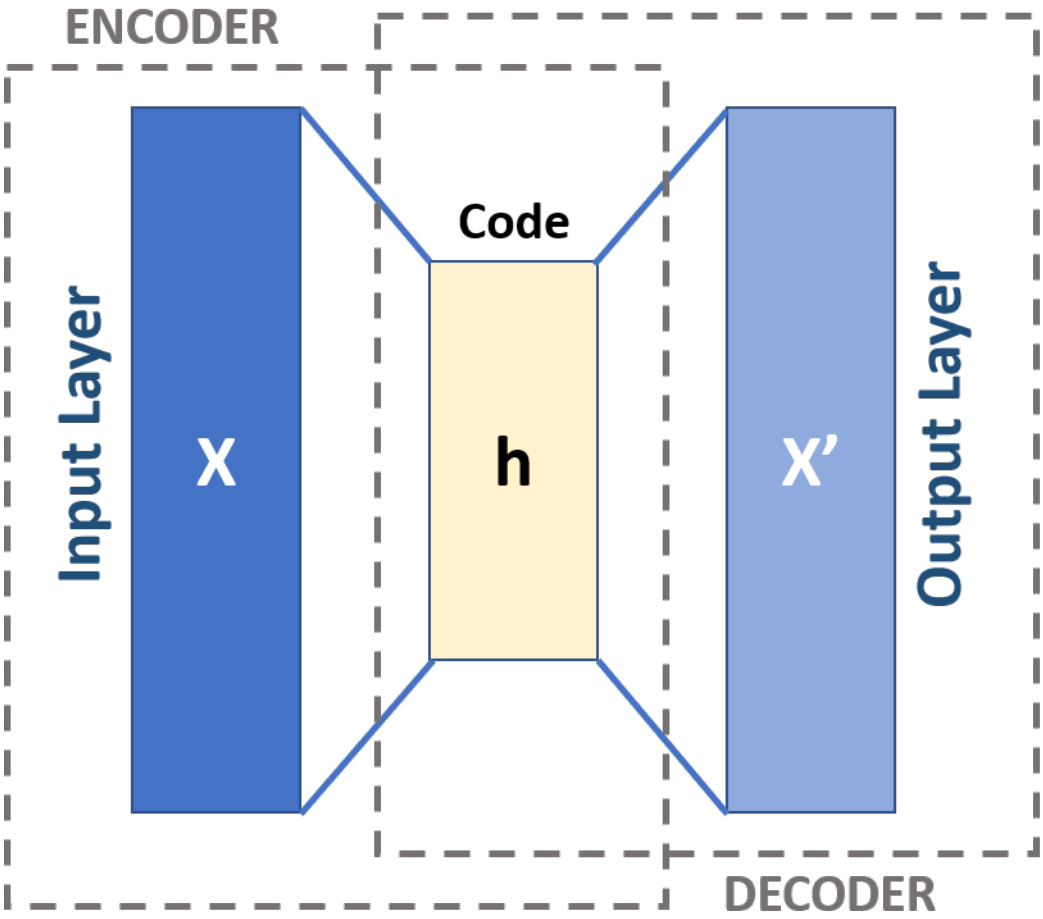

Figure 2.3: **Simple Autoencoder example**

One of the early goals for autoencoders was image compression. By using a lower-dimensional encoding, it could compress an image while keeping the most important features. In the 1990s, autoencoders were also used for feature extraction, in this case, the lower-dimensional encoding could be used as a set of features for neural networks.

However, the performance of basic autoencoders was limited because of the activation functions and optimization algorithms. In the mid-2000s, the rectified linear unit (ReLU) activation function and the stochastic gradient descent optimization algorithm give a resurgence interest in autoencoders. With these, autoencoders could be trained much faster and more efficiently than before.

The variational autoencoder (VAE) became popular in the 2010s. It is a neural network that has gained significant attention because it can generate new data sam-

ples. VAE is a type of model that can learn to generate new samples by training on a dataset of existing samples. VAE can also learn to encode data into a lower-dimensional latent space and then decode it back into the original data space.

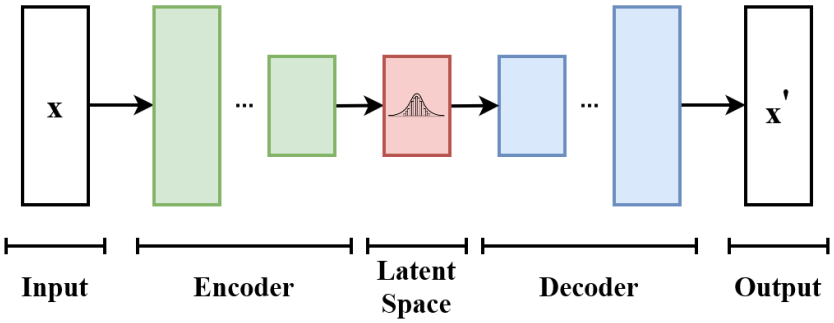

Figure 2.4: **Structure of VAE[54]**

The key difference of VAE is the way in which the latent space is learned. In a basic autoencoder, the encoder maps input data to a fixed-length vector, then decoded back into the original data dimension. However, regarding VAE, the learning process also can involve two main parts. During the encoding, input data is mapped to a probability distribution over the latent space using a neural network. This distribution is typically assumed to be a Gaussian distribution with a mean and variance vector. During the decoding, the sample is drawn from the distribution over the latent space, and this sample is decoded back into the original data dimension using another neural network.

The VAE is trained using a variational lower bound on the log-likelihood of the data. This lower bound is known as the evidence lower bound (ELBO) and is defined as the sum of two terms: the reconstruction error, which measures how well the VAE can reconstruct the input data, and the KL-divergence between the learned distribution over the latent space and a prior distribution over the latent space. The KL-divergence term acts as a regularizer and encourages the learned distribution to be close to the prior distribution.

One of the main benefits of using a VAE is that it allows for the generation of new data samples by sampling from the learned distribution over the latent space. This is achieved by sampling from the Gaussian distribution over the latent space, which can be done efficiently using the reparameterization trick. The reparameterization trick involves sampling from a standard Gaussian distribution and then transforming the

sample using the mean and variance vectors learned by the encoder.[7]

In conclusion, VAE is a powerful generative model that has been used in a wide range of applications, including image generation, music generation, and text generation. The ability to generate new data samples through VAE has significant potential in areas such as art and music, which areas need the creation of new content.

Another autoencoder that has gained popularity is the denoising autoencoder (DAE). DAE has been developed to address the issue of noise in input data. The goal of a traditional autoencoder is to learn a compressed representation of the input data. It can be used for image classification, dimensionality reduction, and anomaly detection. However, when the input data has noise, DAE can learn to simply reproduce the noise rather than capture the structure of data.

DAE was used the solution that incorporated a noise reduction mechanism into the training process to fix this problem. The basic idea is to corrupt the input data with some form of noise before feeding it into the autoencoder. Then trained to reconstruct the original, clean data from the corrupted input.

One of the key benefits of the DAE is its ability to learn robust feature representations that are more resilient to noisy data. By learning to remove the noise from input, DAE is forced to focus on the underlying structure of the data, rather than simply memorizing the noisy patterns. This can lead to improved performance downstream, such as classification or clustering, especially where the input data has inherently noisy.

Another advantage of the DAE is its flexibility with different types of noise. Noise includes Gaussian noise, dropout, and salt-and-pepper noise, but other types of noise can also be used depending on the characteristics of the input data and the task.

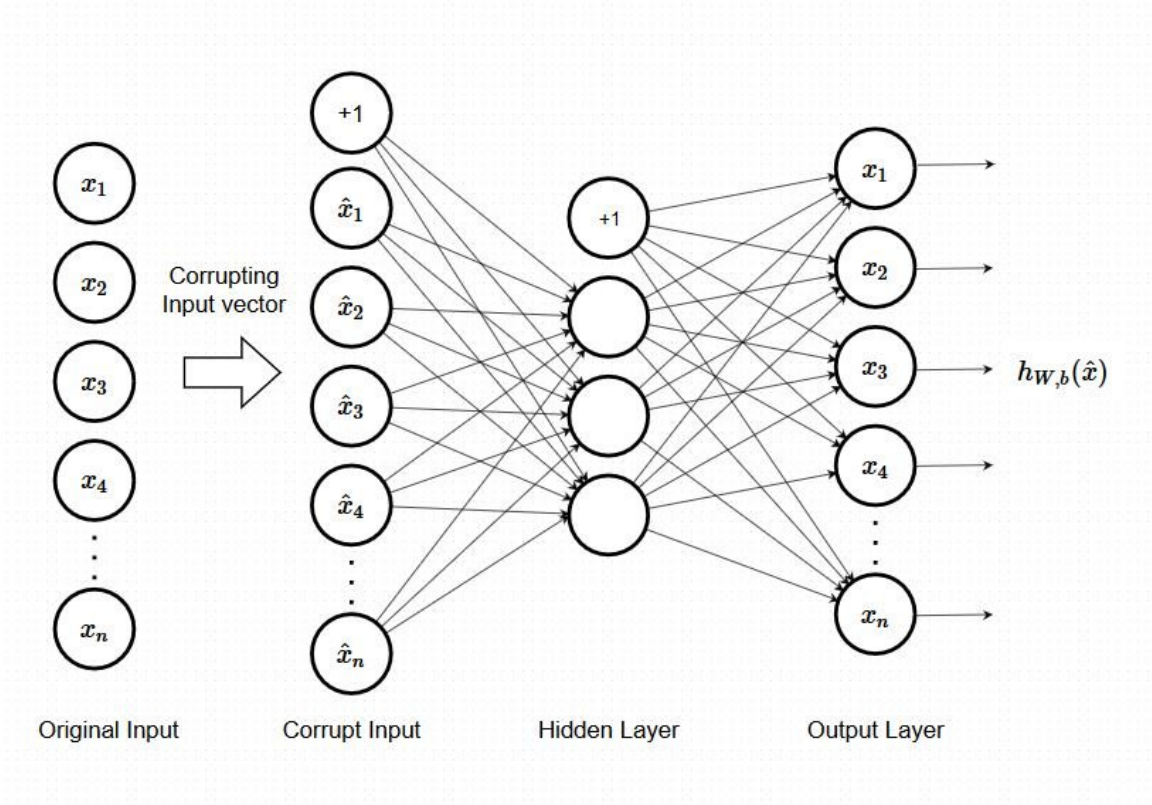

Figure 2.5: **Structure of DAE[55]**

DAE comes with not only benefits but also limitations. One challenge is determining the optimal of noise to add to the input data during training. Less noise may not effectively train the DAE to handle noisy inputs, while too much noise can make it difficult for the model to recover the underlying structure of the data.

In addition, the DAE may not be suitable for all data. For example, in cases where the noise is highly structured and related to the underlying data distribution, DAE maybe struggled to remove it while learning. In this case, adversarial training or generative models may be better suited. It can not be ignored that DAE is a powerful tool in the machine learning toolkit for handling noisy data and learning robust feature representations. [8]

An introduction of these two autoencoders has been shown. Back to the autoencoder itself, which has also been used for unsupervised learning, where the goal is to learn a representation of the data without any labeled examples. By using the reconstruction error as a measure of similarity between data points, it can learn a compressed representation of the data that can be used for clustering and other unsupervised learning goals.

Apparently, autoencoder used to have a long history in the field, and it continues to be active area. With the growth of datasets and computational resources, there are much exciting new using such as computer vision, natural language processing, and robotics.

VGG Network

The VGG network is a deep convolutional neural network that was developed by the Visual Geometry Group (VGG) at the University of Oxford. This network achieved remarkable performance on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2014. The VGG network's success can be attributed to its depth and simplicity, as well as the use of small convolutional filters.[9]

In the early 2010s, deep neural networks became more and more popular in computer vision community. In 2012, Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton developed the AlexNet, which achieved the greatest performance on the ILSVRC. The success of AlexNet sparked a new era, and researchers began to focus on developing deeper and more complex networks.

The VGG network was developed in 2014 and led by Karen Simonyan and Andrew Zisserman. This team aimed to develop a network that could achieve better performance on the ILSVRC while being simpler and more modular than previous.

The VGG network builds with several convolutional layers and fully connected layers. The convolutional layers are organized into blocks, each containing several convolutional layers with the same number of filters and a max pooling layer. The researchers used small filters with a stride of 1 and padding of 1, which allowed having deeper networks without increasing the number of parameters. The use of small filters are also helped the network capture more fine-grained features.

The VGG network was trained on the ILSVRC dataset, which contains over one million images from 1,000 classes. The network was trained using stochastic gradient descent with momentum, with a learning rate schedule that decreased over time. The researchers also used data augmentation techniques, such as random cropping and horizontal flipping, to increase the size of the training set and reduce overfitting.

The VGG network achieved remarkable performance on the ILSVRC, achieving a top-5 error rate of 7.3% on the test set. The network's performance was significantly better than previous, such as AlexNet and ZFNet. And it is also highly modular, which allowed researchers to easily modify and adapt the network for different tasks.

The success of the VGG network paved the way for deeper and more complex convolutional neural networks. Researchers continued to develop larger and more complex networks, such as the ResNet and Inception networks. However, the VGG network remains a popular and influential model in the computer vision community, and its modular architecture and use of small filters continue to inspire new research.

2.1.3 Local Search

The relationship between neural networks and VCD has been confused researchers for many years. NN is a machine-learning algorithm inspired by the structure of biological neurons. They are composed of layers of nodes that are connected by weighted edges and are trained on data to learn a function that maps inputs to outputs. The VCD, is a concept from statistical learning theory that measures the complexity of a hypothesis space, or the set of possible functions that a learning algorithm can choose from.

Research between NN and VCD has focused on understanding how the complexity of a neural network affects its ability to generalize to new data. One of the earliest works in this area was by Valiant in 1984. In his article, he proved a general upper bound on the VC dimension of any function class that can be represented by a feed-forward neural network with a fixed number of hidden units. Since then, researchers took much time to explore the VCD of different types of neural networks, including convolutional neural networks (CNNs), recurrent neural networks (RNNs), and deep neural networks (DNNs).

One approach to analyzing the VCD of NN is to consider about the hypothesis space, in another word, the set of all possible functions that can be represented through network. Understanding the relationship between network architecture and the size of the hypothesis space can provide insights into the VCD. For example, a shallow network with a small number of nodes may have a smaller hypothesis space and lower VC dimension than a deep network with many nodes.

Another approach is to consider the activation function used in the network. The VC dimension can depend on the smoothness and curvature of the activation function, as well as the range of values that it can take. For example, a sigmoid activation function can limit the size of the hypothesis space and reduce the VC dimension compared to a ReLU activation function.

Researchers had also explored overfitting and the VCD. Overfitting occurs when

a model becomes too complex and the training data is not that much, so that can not generalize well to new data. The VCD can provide insights between model complexity and generalization performance. For example, a network with a high VCD may be more prone to overfitting than a network with a lower one, so that regularization can be used to control this.[10][11][12][2][13]

Overall, research between neural networks and VC dimension has contributed to the understanding of the complexity of machine learning models. While there is still much to be explored in this area, the insights gained from this research can inform the design and optimization of neural networks for a variety of applications.

2.2 Related Work

2.2.1 Medical diagnosis

Medical diagnosis applies on specialized knowledge and expertise, which is extremely complex. It needs medical history, physical examination, laboratory tests, and imaging studies for it. In recent years, a growing interest in the use of AI and ML to help having the diagnosis of medical conditions. Also, NN has shown particular promise in medical diagnosis. This part aims to review the literature on the application of neural networks in medical diagnosis.

Medical diagnosis is a particularly promising area for the application of neural networks. The complexity of medical data and the large amount of information that needs to be processed make traditional diagnostic methods need time and afraid of mistakes. Neural networks can process this data more quickly and accurately when having a diagnosis.

Plenty of studies have been conducted on neural networks in medical diagnosis. One of the earliest studies was conducted in 1991 by Storniolo,[23] they used NN to diagnose breast cancer based on mammography images. The conclusion shows that NN can diagnose breast cancer with a sensitivity of 95% and a specificity of 91%.

After this, including heart disease, lung cancer, and neurological disorders, a number of studies have been conducted. In a study conducted by Wang, a neural network can use for processing in diagnosing Alzheimer's disease based on MRI images. This study found that NN was able to accurately diagnose Alzheimer's disease with a sensitivity of 84% and a specificity of 88%.[26]

Another area that has shown excellence is the diagnosis of skin cancer. In a study

conducted by Esteva, a neural network was trained on a dataset of over 130,000 clinical images of skin lesions. And it is able to accurately diagnose skin cancer with an accuracy of 91%, which was comparable to that of expert dermatologists.[25]

In addition to aiding in diagnosis, neural networks have also been used to predict the risk of developing certain medical conditions. A study conducted by Ravì shown that to predict the risk of developing heart disease based on electronic health records,NN can play an important role with the process.[24] This study found that the neural network was able to accurately predict the risk of developing heart disease with an accuracy of 85%.

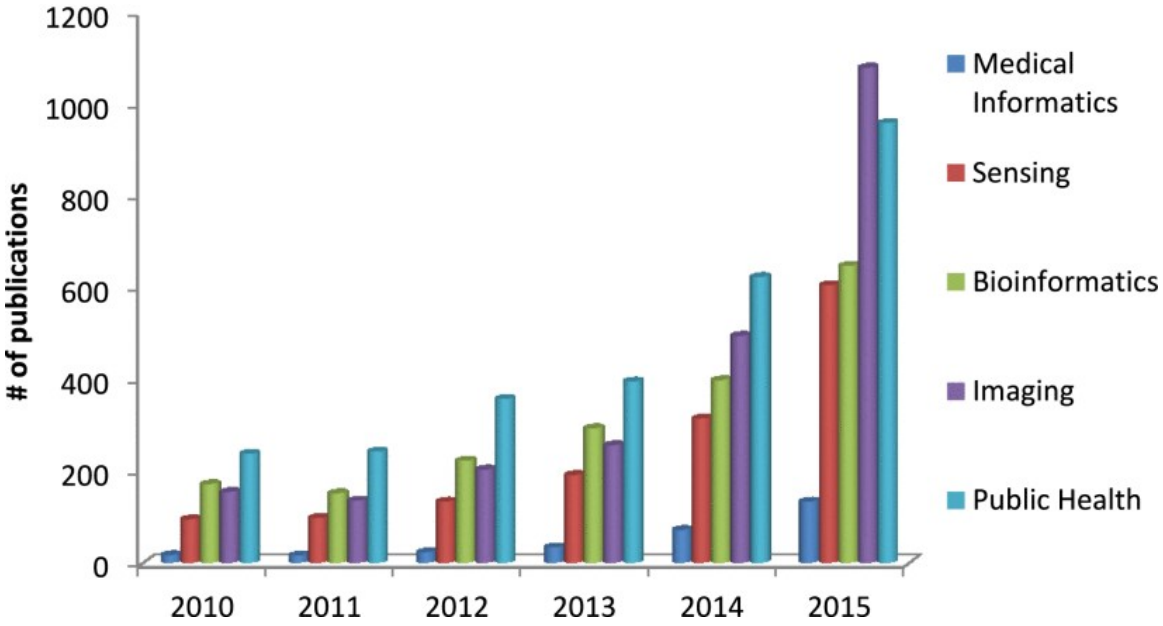

Figure 2.6: **Distribution of published papers that use deep learning in subareas of health informatics. Publication statistics are obtained from Google Scholar in 2017[24]**

The application of NN in medical diagnosis has shown great prospects by improving the speed and accuracy of diagnoses. However, one of the challenges is the need for large datasets for training. This can be particularly challenging in rare medical conditions that may have limited data.

Another challenge is the need for explaining results. Neural networks can be seen as the "black box" that is not always clear on how do results are given. It is also important to understand the reasoning behind a diagnosis, this ca particularly problematic in the medical field.

Although neural networks in medical diagnosis have shown great promise by improving speed and accuracy. While there are still some challenges that need to be addressed, such as the need for large datasets and the interpretability of results, however, the potential benefits are significant.

2.2.2 Language Translation

Language translation is the process of transferring sentence from one language to another language but keeping the meaning same. Because different languages have different syntax, grammar rules, vocabulary, and idioms, this process always shows a big challenge on the computer area. However, in recent years, neural network-based machine translation systems can be trained on huge amounts of bilingual corpora for learning to map between different languages. In this part, a review of the literature on language translation using the neural network will show, and discuss the different architectures and techniques has been used.

The language translation systems are originally based on the encoder-decoder architecture. The encoder takes the sentence needs to be transferred into the source language and encodes it into a fixed-length vector representation. The decoder then takes the encoded vector representation and generates the output sentence in the target language. The training of these neural network models is done using the backpropagation algorithm and the optimization techniques like Stochastic Gradient Descent (SGD), Adam, and Adagrad.

One of the popular neural network-based language translation models is the sequence-to-sequence (seq2seq) model. This model was introduced by Sutskever. in 2014 and has been widely used since then. The seq2seq model consists of two recurrent neural networks, namely the encoder and the decoder. The encoder takes the input sequence and generates a fixed-length vector representation. This vector representation is then used by the decoder to generate the output sequence. The seq2seq model has achieved promising results in machine translation, speech recognition, and image captioning.[27]

Another popular neural network-based language translation model is the transformer model. The transformer model was introduced by Vaswani. in 2017 and has been the state-of-the-art model for machine translation since then. The transformer model uses self-attention to process the input sequence and generate the output sequence. The self-attention mechanism allows the model to attend to different

parts of the input sequence at different positions, enabling it to capture long-range dependencies.[28]

The evaluation of machine translation systems is essential to determine the quality of the output. The most common evaluation metric used for machine translation is the BLEU (Bilingual Evaluation Understudy) score. The BLEU score measures the similarity between the machine-generated output and the reference translation. The BLEU ranges from 0 to 1, higher BLEU better performance. However, it also has some limitations, as it does not consider the semantic meaning and cannot capture the quality for rare words.

In recent years, there have been significant advancements in machine translation systems. One of these is the introduction of the attention mechanism. The attention mechanism allows the model to focus on specific parts of the input sequence while generating the output sequence. This attention mechanism has significantly improved the translation quality of machine translation systems.

Another recent advancement in neural network-based machine translation is the use of pre-trained language models like BERT (Bidirectional Encoder Representations from Transformers) and GPT (Generative Pre-training Transformer). These pre-trained models are trained on massive amounts of text data and can capture the context and meaning of the language better. The use of pre-trained language models has significantly improved the translation quality of machine translation systems.

Neural network-based machine translation systems have made significant progress in recent years, with the transformer model being the state-of-the-art model for machine translation. The evaluation of machine translation systems is done using the BLEU score, but it has some limitations. Recent advancements in neural network-based machine translation include the attention mechanism and the use of pre-trained.

2.2.3 Self-driving

Neural networks are composed of interconnected nodes, or "neurons," that receive inputs from other neurons and produce outputs based on a set of learned weights. By iteratively adjusting these weights in response on training, neural networks can learn to recognize and make predictions with high accuracy. In the self-driving area, neural networks can be used to process sensor data, recognize objects and features, and make decisions about how to drive on the road.

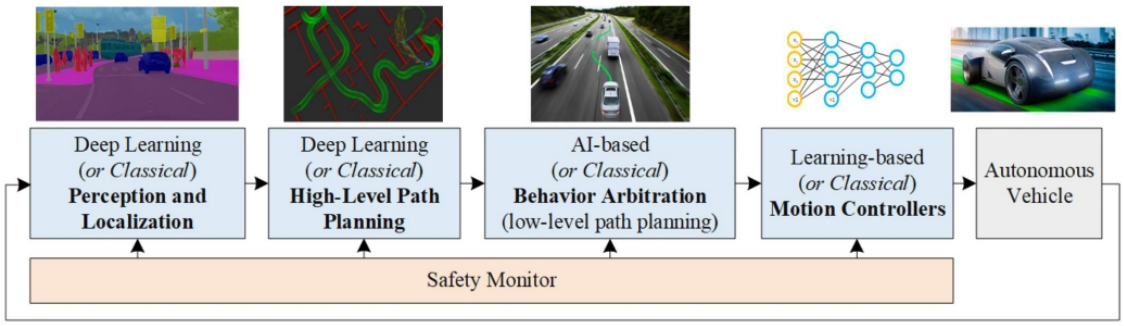

Figure 2.7: **A modular perception-planning-action pipeline[53]**

Early work on self-driving based on NN focused mainly on low-level mission, such as lane detection and obstacle avoidance. For example, in 2014, researchers from NVIDIA developed a deep neural network that could detect road features such as lanes and drivable areas in real time. This is as known as "End-to-End Learning for Self-Driving Cars," trained on a large dataset of images from a front-facing camera on car, and was able to navigate on different kinds of roads and conditions with high accuracy.[29]

Since then,self-driving has expanded to complex mission, such as object detection, pedestrian tracking, and decision-making.

In 2016, Google researchers developed a self-driving car system, which was trained on a large dataset and can detect objects, including pedestrians, cyclists, and other vehicles. By combining with other machine learning algorithms, researchers could develop a fully autonomous driving car that could navigate on both urban and suburban.[30]

Decision-making is also a huge task for self-driving cars. For instance, a team from MIT developed a system called "Social-LSTM" in 2018, which can predict other drivers' behavior. This system was trained on true driving behavior and could accurately predict the movements of other vehicles up to five seconds.[31] By put this predictive ability into a self-driving system, they were able to improve its safety and reliability.

Convolutional neural networks (CNNs) can be used for object detection,[51]which can use filters to identify and make them effective for object recognition tasks. [32] Recurrent neural networks (RNNs)can be used for sequential decision-making, which can used for predicting where a pedestrian might be gone or whether a vehicle wants to change lanes. By analyzing sensor data, self-driving cars can make informed decisions

and respond appropriately to changing road conditions.

In summary, neural networks are a powerful tool in self-driving cars, helping drivers drive safely, and getting the goal of self-dirving.

2.3 Conclusion

In general, for a single definition of VC dimension and neural networks, optimization can be achieved from a mathematical definition perspective. Moreover, mathematical theoretical knowledge has an advanced deductive process for optimizing neural networks. As long as restrictions on specialized terms are completed in the definition, an idealized optimization state can be achieved.

Chapter 3

Analysis of Autoencoders with Vapnik-Chervonenkis Dimension

3.1 Introduction

In this chapter, we exploit VC-dimension techniques to autoencoders for analyzing the optimal network architecture as section 1.5 shows. We obtain a general solution of VC-dimension in equation 3.34 for any Autoencoders. Furthermore, by understanding the statistical principle of VC-dimension in Autoencoder, we have calculated a general solution of VC-dimension to quantify the optimal network structure of Autoencoders. In addition, in the process of calculating the VC-dimension, we also investigate that all the structural factors in Autoencoders significantly influence the VC-dimension limitations including neuron sizes, activation functions, etc. Therefore,

the VC-dimension can be used as an evaluation metric to measure the performance of Autoencoders. There are many factors to consider when building a neural network system. Although It is still mainly used to test the same neural network with different datasets as evaluation, using VC-dimension for performance analysis is a more general and theoretical manner.

3.2 Related Work

In the past decades, many important results have been proposed in the field of VC-dimension to analyze several specific behaviors in neural networks. In 1989, Abu[39] indicated that VC-dimension could use as a theoretical measurement to quantify the learning capacity of neural networks. These years, using VC-dimension for evaluation is also a topic that has been studied. Bartlett[43] and Pinto[44] select sample complexity as assessed by the VC dimension. And transform the data by increasing the dimension of the input features based on the sample complexity evaluated by the VC dimension. Chen [38] provided theoretical insights that SVM is actually designed from both VC-dimension theory and the principle of structural risk minimization, obtaining better generalization performance with small, non-linearity, high-dimensionality samples.

3.2.1 Statistics concepts on VC-Dimension

It is well known that two of the most important aspects of machine learning models are how well the model generalizes to unknown data, and how well the model scales with problem complexity[35]. For a neural network, the influence of variables may sometimes exceed the architecture of the entire neural network, which is the number of hidden layers, the number of neurons, the weights, and the activation function. Moreover, overfitting is a fundamental issue in supervised machine learning which prevents us from perfectly generalizing the models to well-fit observed data on training data, as well as unknown data on the testing set. Overfitting occurred[36] because of the presence of noise, the limited size of training sets, and complexity of classifiers.

Here we firstly introduce several concepts, assuming that the hypothesis space as H, $\hat{Y}$ is the ideal output of the model, and Y is the actual output. The expected error is E_X ,the empirical error is E_M . The goal is to make $\hat{Y}$ approximately equal to Y, and $E_M(\hat{Y}) = 0$. Which means that $E_M(Y) \approx 0$. The Hoeffding inequality

that must be mentioned first[37]:

Definition 1. *For a group of independent random variables $X_1, ..., X_n \in \mathbb{R}$, assuming for all $a_i \leq i \leq b_i$, which is*

$$\mathbb{P}(X_i \in [a_i, b_i)) = 1 \tag{3.1}$$

The sum of random variables is:

$$S_n = X_1 + ... + X_n \tag{3.2}$$

The expected value of S_n is $E(S_n)$ So for all $t \geq 0$:

$$\mathbb{P}(|S_n - E(S_n| \geq t) \leq 2exp(-\frac{2t^2}{\sum_{i=1}^{n}(b_i - a_i)^2}) \tag{3.3}$$

For one hypothesis h in H, when the number of samples N is large enough, use the Hoeffding inequality to infer the overall expected error $E_X(h)$ through the empirical error $E_M(h)$ on the sample set:

$$\mathbb{P}[|E_X(h) - E_M(h)| > \epsilon] \leq 2exp(-2\epsilon^2 N) \tag{3.4}$$

So that, when N is large enough, $E_X(h)$ will be close enough to $E_M(h)$. This situation only suit for only one hypothesis in H. Now let us assumed that there are M hypothesises in H, which is $h_1, h_2..., h_M$, $E_[h_i] = |E_X(h) - E_M(h)|$. The Hoeffding inequality will be:

$$\begin{aligned} &\mathbb{P}[|[E_{h1}| > \epsilon \cup |E_{h2}| > \epsilon \cup ... \cup |E_{hM}| > \epsilon] \\ &\qquad \leq \mathbb{P}[E_{h1} > \epsilon] + \mathbb{P}[E_{h2} > \epsilon] + ... + \mathbb{P}[E_{hM} > \epsilon] \\ &\qquad \leq 2Mexp(-2\epsilon^2 N) \end{aligned} \tag{3.5}$$

It can be rewritten as:

$$\forall\ Y \in H,\ \ \mathbb{P}[|E_X(Y) - E_M(Y)| > \epsilon] \leq 2Mexp(-2\epsilon^2 N) \tag{3.6}$$

The conclusion here is that the number of samples needs to be large enough under the assumption that the number M is finite. If the number of hypotheses M in the hypothesis space is infinite, then the limit $2Mexp(-2\epsilon^2 N)$ will also become infinite,

which means that learning is meaningless. So equation (3.6) will be:

$$\forall\ Y \in H,\ \ \mathbb{P}[|E_X(g) - E_M(g)| > \epsilon] \leq 2eff(M)exp(-2\epsilon^2 N) \tag{3.7}$$

In order to define a finite M, and get rid of dataset(No longer limited to any one particular dataset), a growth function need to be added[46][47][48],

$$m_H(N) = max_{X_1,X_2,...,X_N \in X}|H(X_1, X_2, ..., X_N)| \tag{3.8}$$

The growth function's superior bound is 2^N, So that M changed from limit to 2^n, to reduce the magnitude, we need to introduce break point:

Definition 2. *For the growth function $m_H(N)$ of the hypothesis space H, N is the sample size. When $N = k$,$m_H(N) < 2N$, k is the break point of H.*

Thus if the break point is available, growth function $m_H(N)$ will be a polynomial, and the magnitude will reduce, which means the learning is meaningful. Using both a break point and the growth definition, we could change the equation 3.7 of VC bound as:

$$\forall\ Y \in H, \mathbb{P}[|E_X(Y) - E_M(Y)| > \epsilon] \leq 4M_H(2N)exp(-\frac{1}{8}\epsilon^2 N) \tag{3.9}$$

It shows that, as N gradually increases, the exponential $exp(\cdot)$ decreases faster than the polynomial $M_H(2N)$ increases. According to this, we get the definition of the VC-dimension on the hypothesis space H[47]:

Definition 3. *Suppose the VC-dimension of space H is the size of the largest dataset that can be broken up by H, that is:*

$$VC(H) = max\{N : m_H(N) = 2^N\} \tag{3.10}$$

So, the $VC(H) = k - 1$, k is the break point of H.

We know that the VC-dimension did not connect with the learning algorithm, the specific distribution of the dataset or the objective function. It is only influenced by the model itself and hypothesis space.

3.2.2 Single-layer Network with VC-dimension

The traditional definition for the VC-dimension is for an indicator function set. If there are H samples that can be separated by the functions in the function set in all

possible forms of the H power of 2, then the function set is said to be able to break up the H samples.; The VC-dimension of the function set is the maximum number of samples H that it can break up. As for the connection between VC-dimension and neural networks, I have to mention the article published by Sontag in 1998. First, he pointed out that the VC-dimension is oriented toward binary classification. The concept of VC-dimension can be generalized in a number of ways to deal with the problem of "learning" (approximating from data) real-valued functions. This also leads to pseudo-dimensions, fat-crushing dimensions, and several other concepts[33]. In his paper, he assumed that a set U, which has been called as the input space, U is also a subset of R^m. The definition of VC-dimension in Neural Networks has been provided as:

Definition 4. *If F is a vector subspace of R^U, then VC-dimension of F =dimF*

This is directly applied to the perceptron[33] , which is just a linear discriminator that exists in R^m,and its VC-dimension is defined as:

$$VCDP_m = m + 1 \tag{3.11}$$

When this definition is directly applied to Single Hidden Layer Nets with Fixed Input Weights, it becomes different.
Here is a defined single hidden layer neural network shown in Figure 1,it has been defined as an n row,m vectors,input-layer weight A_1,...,A_m, input-layer bias b_1,...,b_n, output-layer weight C_0,...,C_m, $\sigma(A_i u + B_i), i = 1, ..., n$, so the dimension will be[33]:

$$VCDF_{n,\sigma,A,B} \leq n + 1 \tag{3.12}$$

The conditions under which the above equations hold are related to the choice of activation function. This article exemplifies when tanh is selected as the activation function, and, (A_i,b_i) $\neq \pm$(A_j,b_j) for all i $\neq$ *j and that* $A_i \neq 0$ *for all i* [33]. Here, the calculation of the VC-dimension of the neural network can be extended to more complex neural networks, such as Autoencoders, which can automatically generate weights and biases[34]. It is still important to note that the most important point in Sontag's theory is that in this proposition[33, 34], different constraints need to be matched with different activation functions before the corresponding VC-dimension can be calculated.

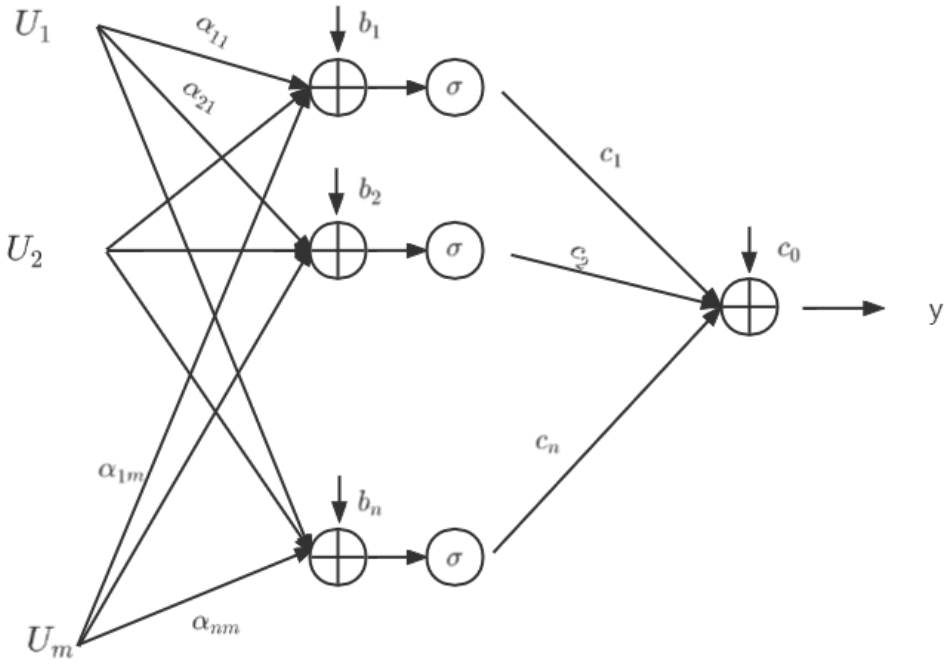

Figure 3.1: Single-hidden layer neural net

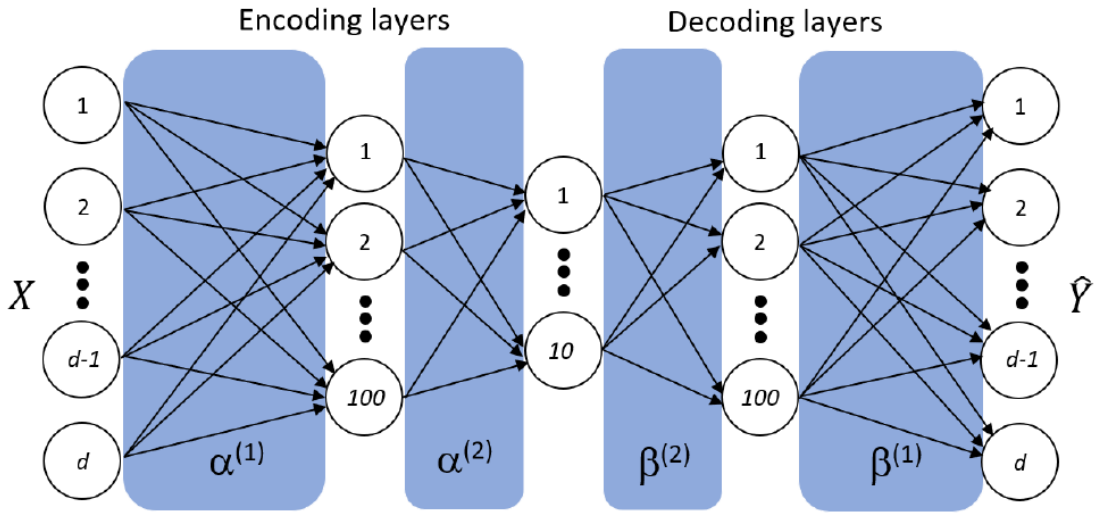

Figure 3.2: The Autoencoder model [56]

3.3 VC-dimension in Autoencoders

For Autoencoders, it is more efficient to compute the VC-dimension separately for the encoder and decoder. When discussing the impact of VC-dimension on Autoencoders, the selected neural network architecture needs to be mentioned. Therefore, even the number of neurons, the number of hidden layers, and other variables are all the same, the difference in activation functions will also lead to the difference in the VC-dimension and the VC bound. Therefore, reducing all the constant factors, and simply calculating the different representations of VC bounds caused by different activation functions has become the first topic to be discussed in this section. We first simplify the structure of the Autoencoder structure as mentioned in Fig. 2, then will extend the solution to a general network architecture of Autoencoder at the end.

3.3.1 VC dimension of known Autoencoders

This Autoencoder has its input X, target Y and the neutral network output $\hat{Y}$.The dimension of $X, Y, \hat{Y}$ will be d_0. In Section 3.2.1, we learned that if the superior limit of the number of hypotheses in the hypothesis space can be calculated, it will be more possible to calculate the VC-dimension. The VC-dimension is a measure of the capacity (complexity, expressiveness, richness, or flexibility) of the space of functions that can be learned by statistical classification algorithms. Therefore, for

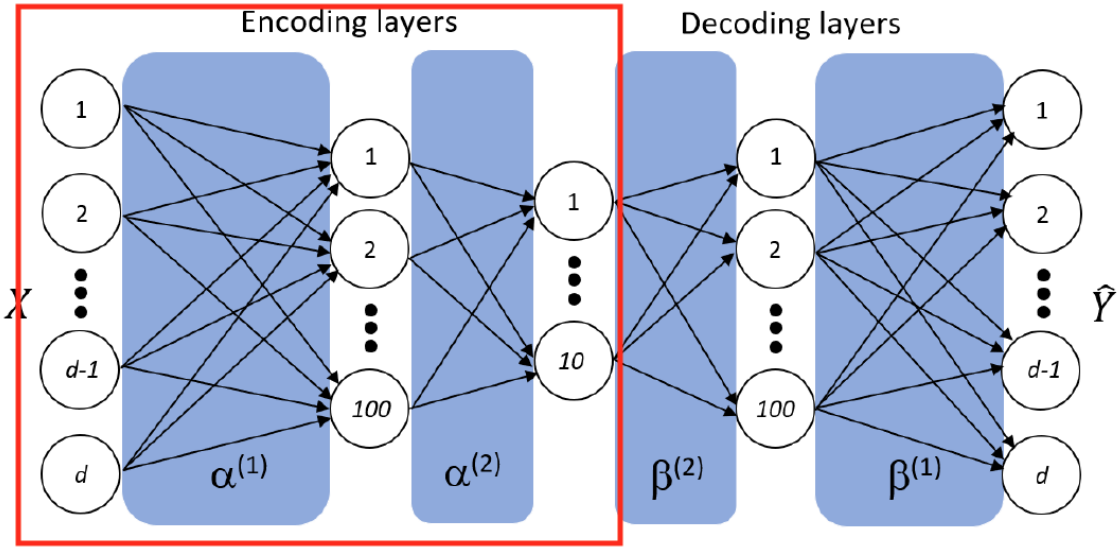

Figure 3.3: Encoding layers[56]

the VC-dimension, the space complexity will affect the value to a certain extent, but this calculation is based on the same conditions as other variables. However, the main comparison in this article is in the encoder/decoder part, how will the VC-dimension change, and to what extent will it be affected when the selected activation function is different. The two activation function that will be used is the sigmoid function[49]:

$$g(x) = \frac{1}{1 + e^{-x}} \tag{3.13}$$

And the hyperbolic tangent function:

$$g(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{3.14}$$

First divide this Autoencoder into two parts, encoder and decoder. Given a training dataset, the dimensionality of the original data (d) will be progressively reduced to d_1, and d_2 through the encoding layers, then will be increased to d_3, and d respectively. For the encoder layer, X_{11},X_{21},...,X_{n1} is the input data, the dimension will be d_0. Y_{11},Y_{21},...,Y_{n1},is on the first layer which will reduced to d_1 dimensions, the bias is b_1. Then Y_{12},Y_{22},...,Y_{n2} has the d_2 dimensions, the bias is b_2, σ as the activation function.

Theorem 1. *For this Autoencoder encoder part, the VC-dimension will be:*

$$VCD_{f_{\sigma,b,X,Y_{n2}}} \leq d_1^2 + 2 \tag{3.15}$$

Proof. z_1 as the first layer, so :

$$z_1 = \sigma(W_{Xn1} + b_1) \tag{3.16}$$

z_2 as the second layer:

$$z_2 = \sigma(W_{Yn1} + b_2) \tag{3.17}$$

Make X' as the output of the encoding layers, which is:

$$X' = \sigma(W_{Yn1}(\sigma(W_{Xn1} + b1) + b_2) \tag{3.18}$$

It can be written as:

$$X' = \sigma^2 W_{Yn1} \cdot W_{Xn1} + \sigma^2 W_{Yn1} \cdot b_1 + \sigma b_2 \tag{3.19}$$

Introduce the definition of space complexity that is particular in this neural network

Definition 5. *Space complexity can be shown as:*

$$Space \sim O(\sum_{l=1}^{D} K_l^2 \cdot C_{(l-1)} \cdot C_l) \tag{3.20}$$

For the space complexity, the theory is usually used in convolutional neural networks, which also means that the space complexity affects the VC-dimension, while the sample size does not affect the space complexity. For this, the dimension for the first layer is d_1,so $O(Y_{n1}) = d_1^2$. Therefore, since we know that the essence of encoding is to reduce the dimension, the dimension of the final output X' is d_2, which means $O(X') = d_2$, much lower than $O(Y_{n1})$. So for the encoding part, the $O(X') = d_1^2$.
Based on (3.19), which is a multivariate quadratic equation, it let the VC-dimension becomes:

$$VCD_{f_{\sigma,b,X,X'}} \leq d_1^2 + 2 \tag{3.21}$$

□

When can the equation holds, and the VC-dimension reaches the maximum value, which is $d_1^2 + 2$?

Theorem 2.

$$VCD_{f_{\sigma,b_m,X_n,Y_{n3}}} = d_1^2 + 2, when\ b_2 = 0\ while\ \sigma = sigmoid \neq 0$$

Proof. When the activation function is $\sigma = sigmoid$, the equation (3.19) becomes:

$$\sigma^2 W_{Yn1} \cdot W_{Xn1} + \sigma^2 W_{Yn1} \cdot b_1 + \sigma \cdot b_2 = 0 \tag{3.22}$$

Assume $\sigma \neq 0$:

$$\sigma \cdot W_{Yn1} \cdot W_{Xn1} + \sigma \cdot W_{Yn1} \cdot b_1 + b_2 = 0 \tag{3.23}$$

Since $W_{Yn1} = (W_{Xn1} + b_1)$, so

$$\sigma(W_{Xn1} + b_1) \cdot W_{Xn1} + \sigma(W_{Xn1} + b_1) \cdot b_1 + b_2 = 0 \tag{3.24}$$

Take b_2 into another side:

$$W_{Xn1}^2 + 2 \cdot W_{Xn1} b_1 + b_1^2 = -b_2 \cdot (1 + e^{-x}) \tag{3.25}$$

So that it can be seen like:

$$(W_{Xn1} + b_1)^2 = -b_2 \cdot (1 + e^{-x}) \tag{3.26}$$

The left side of the equation becomes a quadratic polynomial, which means that the value of the left side of the equation is ≥ 0. On the right side of the equation, there are $(-b_2)$ and the sigmoid function. The value of the sigmoid function is (0,1). It is assumed that the sigmoid function is not zero, so the only way to equal is that $b_2 = 0$. □

What if the σ changed? The tanh function also is a popular option as an activation function. It cannot be ignored that the value of tanh function is $[-1, 1]$. It is worth noting that this is an important difference from sigmoid function so:

Theorem 3. *When* $(W_{Xi1}, b_{i1}) \neq \pm(W_{Xj1}, b_{j1})$ *for all* $i \neq j$ *, and* $b_{j2} \neq 0$ *for all* j *while* $\sigma = tanh \neq 0$ *in the encoder part, the VC-dimension will be:*

$$VCD_{f_{\sigma,b_m,X_n,Y_{n3}}} = d^2 + 2 \tag{3.27}$$

The proof process can be found in Sontag[33]. It can be seen from these two

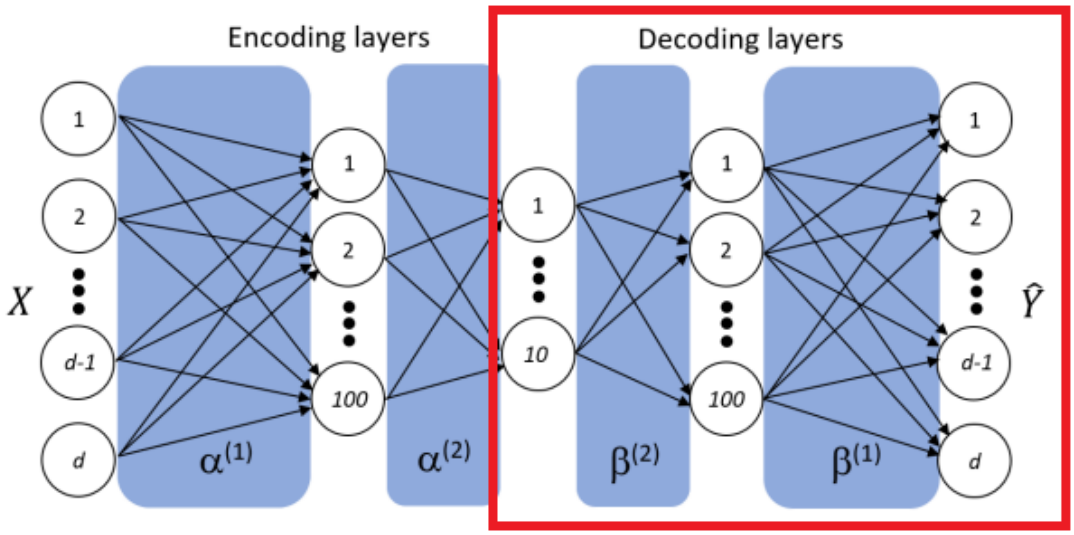

Figure 3.4: Decoding layers

different activation functions, the constraints on reaching the upper limit of the VC-dimension are different.

After the derivation of the encoder, for the VC-dimesion of the decoder, the factor that does not need to be considered is the activation function. In the usual sense, the decoder uses the inverse function of the encoder as the activation function, which is σ'. For decoder, X'_1,X'_2,...,X_n is the input data,the dimension will be d_2. Y_{13},Y_{23},...,Y_{n3},is the the first decoding layer which will increase to $d_3 = d_1$ dimensions, the bias is b_3. Then Y_{14},Y_{24},...,Y_{n4} has the d_0 dimensions, the bias is b_4. The $\hat{Y}$, which is the output for the decoding part, will be:

$$\hat{Y} = \sigma'(W_{Y_{n4}}(\sigma'(W_{X'} + b3) + b_4) \tag{3.28}$$

The space complexity will change. Because the structure of decoding is different from encoding, 10 neurons in the first layer of decoding will be added to the space complexity, which is:

$$O'(\hat{Y}) = d_1^2 + d_2 \tag{3.29}$$

Theorem 4. *So the VC-dimension of decoding will be:*

$$VCD_{f_{\sigma',b_m,X',\hat{Y}}} \leq d_1^2 + d_2 + 2 \tag{3.30}$$

When $\sigma = sigmoid$, through Theorem 2,

$$VCD_{f_{\sigma,b_m,X',\hat{Y}}} = d_1^2 + d_2, when\ b_4 = 0\ while\ \sigma = sigmoid \neq 0 \tag{3.31}$$

While the activation function is tanh, when $(W_{Yi2}, b_{i3}) \neq \pm(W_{Yj2}, b_{j3})$for all $i \neq j$, and $b_{j4} \neq 0$ for all j while $\sigma = tahh \neq 0$, the VC-dimension will be:

$$VCD_{f_{\sigma',b_m,X',\hat{Y}}} \leq d_1^2 + d_2 \tag{3.32}$$

So, for this whole Autoender, the VC-dimension will be:

$$VCD_{f_{\sigma',b_m,X',\hat{Y}}} \leq d_1^2 + d_2 + 2 \tag{3.33}$$

3.3.2 VC dimension of Autoencoders with unfixed structure

When the number of layers and the number of neurons in the hidden layer are unknown, how to calculate the VC-dimension becomes to the extended talks. We use an unfixed Autoencoder structure where $N_1, N_2, ..., N_n$ are the dimension of each hidden layer, M is the number of encoding layer, X denotes input data, Y denotes output data, σ is the activation function, and $b_1, b_2, ..., b_{2M}$ are the bias for each layer. Through Theorem 4, the VC-dimension will be:

$$VCD_{f_{\sigma',b_m,X,Y}} \leq N_1^M + N_2^{M-1} + ... + N_n^1 + M \tag{3.34}$$

When the activation function is sigmoid function, it will have a limitation, which is: $b_M, b_{2M} \neq 0\ while\ \sigma \neq 0$. If σ changed as tanh, the limitation is: $(W_{YiM}, b_{i(M+1)}) \neq \pm(W_{YjM}, b_{j(M+1)})$ for all $i \neq j$, and $b_{j(2M)} \neq 0$ for all j.

3.4 Conclusion

In this part, the VC-dimension on Autoencoders has been provided. In the case of choosing different activation functions, there will be other constraints. But in essence, for the Autoencoder, selecting the appropriate number of neurons for the hidden layer becomes the best way to optimize itself. When neuron number reaches a peak, the learning ability of the neural network declines, so selecting a suitable activation function and the number of neurons has become a way to optimize the

VC-dimension. Therefore, these provided results could be used to solve the optimization problem. VC-dimension still has a massive impact on neural network systems. Through calculation, we know that when the space complexity of a nervous system is determined, the analysis of VC-dimension is obvious. However, this paper doesn't cover any VC-dimension results for deep networks. It is worth further investigating the VC-dimension for the deep network such as deep convolutional neural networks.

Chapter 4

VC-dimension in VGG Network

4.1 Overview

In the preceding chapter, we delved into the impact of various activation functions on the optimal performance of the Autoencoder. This chapter extends our inquiry by exploring the influence of the number of hidden layers on the VC-dimension of deep learning models. As section 1.5 shown, specifically, this chapter investigates the VC dimension of two comparable models, VGG16 and VGG19, and examines how the VC dimension is affected by the number of hidden layers, while holding constant

other variables such as activation function and input data. It also provided a solution for how to compute VC dimension in a 2D convolution layer. Our objective is to demonstrate that as the number of hidden layers increases, the VC dimension can support the model more efficiently from a mathematical perspective. To achieve this, we perform mathematical calculations to derive the VC dimension of each model and validate our results through experimental verification. The findings of this chapter contribute to a better understanding of the relationship between the VC dimension and the number of hidden layers in deep learning models.

4.2 Related Work

The VGG network, a deep convolutional neural network architecture, was first introduced by the renowned research group, VGG at the University of Oxford, in the year 2014. Its exceptional performance in the Localization Task, where the task was to accurately locate objects in an image, and the second-place ranking in the Classification task, where the objective was to classify objects in images, at the ImageNet competition that year, established it as a significant milestone in the field of computer vision.

For the convenience of calculation, in this chapter it is recorded as $M_{in} \times N_{in} \times R_{in}$, and the default activation function all is ReLu function. After computing the output of the entire network, the space complexity of the network is given. Finally, different VC dimensions of different networks are given. The size formula for the output image of the convolutional layer is as follows, the W_{filter} and H_{filter} is the width and height of the filter respectively. P is the number of boundary pixel layers filled at the edge of the image. S is for Stride.

$$W_{out} = \frac{W_{in} - W_{filter} + 2P}{S} + 1 \tag{4.1}$$

$$H_{out} = \frac{H_{in} - H_{filter} + 2P}{S} + 1 \tag{4.2}$$

For maxpool layers, the size calculation of the output image will use two different formula, which is:

$$W_{out} = \frac{W_{in} - W_{filter}}{S} + 1 \tag{4.3}$$

$$H_{out} = \frac{H_{in} - H_{filter}}{S} + 1 \tag{4.4}$$

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input(224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Table 4.1: The VGG Network

In addition, the sliding window size of the convolution kernel is unified as 3×3 , and the step size is unified as 1.

In this section, we will discuss the space complexity function of the convolutional layer, which is distinct from the space complexity function of the autoencoder discussed in the previous chapter. This is due to the fact that the VGG16 network employs a greater number of convolutional layers for computation, as opposed to the simpler Autoencoder architecture. The space complexity function refers to the amount of space required to execute a particular algorithm, and in the case of the

VGG16 network, it is related to the number of parameters used in the convolutional layers. The more parameters used in the convolutional layers, the greater the space complexity of the VGG16 network. This distinction is important to consider when evaluating the computational efficiency of the VGG16 network and its suitability for different applications. K as the size of the convolution kernel, C is the number of channels, and D is the number of layers. So the space complexity also needs to be calculated again, which is shown as:

Definition 6. *Space complexity can be calculated as:*

$$Space \sim O(\sum_{l=1}^{D} K_l^2 \cdot C_{(l-1)} \cdot C_l + \sum_{l=1}^{D} M^2 \cdot C_l) \tag{4.5}$$

4.3 VC-dimension of VGG16

For the purpose of calculating and analyzing the VGG16 network, it has been partitioned into six distinct blocks. When inputting image data, the first and second hidden layers, namely conv3-64, along with the first layer maxpool, constitute block 1. Block 2 comprises of two conv3-128 layers and the second maxpool layer. Similarly, block 3 is composed of three conv-254 layers and the third maxpool layer. The fourth block consists of three conv3-512 layers and the fourth maxpool layer. Block 5 includes the next three conv3-512 layers and the fifth maxpool layer. Finally, block 6 encompasses the last three fully connected layers and the output layer, which will be computed together for further analysis.

4.3.1 Block 1

input(224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					

Table 4.2: Block 1 in VGG16

Input layer: the input image will be $M_{in} \times N_{in} \times R_{in}$.
Conv3-64, padding is 1, so:

$$W_{out} = \frac{W_{in} - W_{filter} + 2P}{S} + 1 = \frac{M_{in} - 3 + 2}{1} + 1 = M_{in} \tag{4.6}$$

$$H_{out} = \frac{H_{in} - H_{filter} + 2P}{S} + 1 = \frac{N_{in} - 3 + 2}{1} + 1 = N_{in} \tag{4.7}$$

The known depth is 64, so the feature dimension of the output feature map is $M_{in} \times N_{in} \times 64$.

Again Conv3-64, padding is 1, so:

$$W_{out} = \frac{W_{in} - W_{filter} + 2P}{S} + 1 = \frac{M_{in} - 3 + 2}{1} + 1 = M_{in} \tag{4.8}$$

$$H_{out} = \frac{H_{in} - H_{filter} + 2P}{S} + 1 = \frac{N_{in} - 3 + 2}{1} + 1 = N_{in} \tag{4.9}$$

The feature dimension of the output feature map is still $M_{in} \times N_{in} \times 64$.

Turn to the first maxpool layer, the sliding window size is 2×2, and the step size is also 2. This pooling layer sliding window dimension $2 \times 2 \times 64$. So:

$$W_{out} = \frac{W_{in} - W_{filter}}{S} + 1 = \frac{M_{in} - 2}{2} + 1 = \frac{M_{in}}{2} \tag{4.10}$$

$$H_{out} = \frac{H_{in} - H_{filter}}{S} + 1 == \frac{N_{in} - 2}{2} + 1 = \frac{N_{in}}{2} \tag{4.11}$$

The feature dimension of the output feature map in Block 1 is $\frac{M_{in}}{2} \times \frac{N_{in}}{2} \times 64$.

For the space complexity calculation of Block 1, which is $144M_{in}^2 + 53824$, equation is as below:

$$\begin{aligned} Space &\sim O(\sum_{l=1}^{D} K_l^2 \cdot C_{(l-1)} \cdot C_l + \sum_{l=1}^{D} M^2 \cdot C_l) \\ &= (3^2 \cdot R_{in} \cdot 64 + 3^2 \cdot 64 \cdot 64 + 2^2 \cdot 64 \cdot 64) \\ &\quad + (M_{in}^2 \cdot 64 + M_{in}^2 \cdot 64 + (\frac{M_{in}}{2})^2 \cdot 64) \end{aligned} \tag{4.12}$$

4.3.2 Block 2

input(224 × 224 RGB image)					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					

Table 4.3: Block 2 in VGG16

First layer is Conv3-128, the padding is 1, so:

$$W_{out} = \frac{W_{in} - W_{filter} + 2P}{S} + 1 = \frac{\frac{M_{in}}{2} - 3 + 2}{1} + 1 = \frac{M_{in}}{2} \tag{4.13}$$

$$H_{out} = \frac{H_{in} - H_{filter} + 2P}{S} + 1 = \frac{\frac{N_{in}}{2} - 3 + 2}{1} + 1 = \frac{N_{in}}{2} \tag{4.14}$$

So the feature dimension of the output feature map is $\frac{M_{in}}{2} \times \frac{N_{in}}{2} \times 128$.
Next layer is still Conv3-128, same as the first layer in Block 2, so the feature dimension of the output feature map still $\frac{M_{in}}{2} \times \frac{N_{in}}{2} \times 128$. For the second maxpooling layer,

$$W_{out} = \frac{W_{in} - W_{filter}}{S} + 1 = \frac{\frac{M_{in}}{2} - 2}{2} + 1 = \frac{M_{in}}{4} \tag{4.15}$$

$$H_{out} = \frac{H_{in} - H_{filter}}{S} + 1 == \frac{\frac{N_{in}}{2} - 2}{2} + 1 = \frac{N_{in}}{4} \tag{4.16}$$

The feature dimension of the output feature map in Block 2 is $\frac{M_{in}}{4} \times \frac{N_{in}}{4} \times 128$.
For the space complexity calculation of Block 2, which is $72M_{in}^2 + 286720$, equation is as below:

$$\begin{aligned} Space &\sim O(\sum_{l=1}^{D} K_l^2 \cdot C_{(l-1)} \cdot C_l + \sum_{l=1}^{D} M^2 \cdot C_l) \\ &= (3^2 \cdot 64 \cdot 128 + 3^2 \cdot 128 \cdot 128 + 2^2 \cdot 128 \cdot 128) \\ &+((\frac{M_{in}}{2})^2 \cdot 128 + (\frac{M_{in}}{2})^2 \cdot 128 + (\frac{M_{in}}{4})^2 \cdot 128) \end{aligned} \tag{4.17}$$

4.3.3 Block 3

input(224 × 224 RGB image)					
conv3-256	conv3-256	conv3-256	conv3-256	conv3-256	conv3-256
conv3-256	conv3-256	conv3-256	conv3-256	conv3-256	conv3-256
			conv1-256	**conv3-256**	conv3-256
					conv3-256
maxpool					

Table 4.4: Block 3 in VGG16

Block 3 has 3 Conv3-256 layer and 1 maxpool layer, so the feature dimension of the output feature map for the third Conv3-256 will be $\frac{M_{in}}{4} \times \frac{N_{in}}{4} \times 256$. For the third

maxpool layer, it will be:

$$W_{out} = \frac{W_{in} - W_{filter}}{S} + 1 = \frac{\frac{M_{in}}{4} - 2}{2} + 1 = \frac{M_{in}}{8} \tag{4.18}$$

$$H_{out} = \frac{H_{in} - H_{filter}}{S} + 1 == \frac{\frac{N_{in}}{4} - 2}{2} + 1 = \frac{N_{in}}{8} \tag{4.19}$$

The feature dimension of the output feature map in Block 3 is $\frac{M_{in}}{8} \times \frac{N_{in}}{8} \times 256$. For the space complexity calculation of Block 3, which is $52M_{in}^2 + 1736704$, equation is as below:

$$\begin{aligned} Space &\sim O(\sum_{l=1}^{D} K_l^2 \cdot C_{(l-1)} \cdot C_l + \sum_{l=1}^{D} M^2 \cdot C_l) \\ &= (3^2 \cdot 128 \cdot 256 + 3^2 \cdot 256 \cdot 256 + 3^2 \cdot 256 \cdot 256 + 2^2 \cdot 256 \cdot 256) \\ &+((\frac{M_{in}}{4})^2 \cdot 256 + (\frac{M_{in}}{4})^2 \cdot 256 + (\frac{M_{in}}{4})^2 \cdot 256 + (\frac{M_{in}}{8})^2 \cdot 256) \end{aligned} \tag{4.20}$$

4.3.4 Block 4

input(224 × 224 RGB image)					
conv3-512	conv3-512	conv3-512	conv3-512	conv3-512	conv3-512
conv3-512	conv3-512	conv3-512	conv3-512	conv3-512	conv3-512
			conv1-512	**conv3-512**	conv3-512
					conv3-512
maxpool					

Table 4.5: Block 4 in VGG16

Block 4 has 3 Conv3-512 layer and 1 maxpool layer, so the feature dimension of the output feature map for the third Conv3-512 will be $\frac{M_{in}}{8} \times \frac{N_{in}}{8} \times 512$. For the fourth maxpool layer, it will be:

$$W_{out} = \frac{W_{in} - W_{filter}}{S} + 1 = \frac{\frac{M_{in}}{8} - 2}{2} + 1 = \frac{M_{in}}{16} \tag{4.21}$$

$$H_{out} = \frac{H_{in} - H_{filter}}{S} + 1 == \frac{\frac{N_{in}}{8} - 2}{2} + 1 = \frac{N_{in}}{16} \tag{4.22}$$

The feature dimension of the output feature map in Block 4 is $\frac{M_{in}}{16} \times \frac{N_{in}}{16} \times 512$. For the space complexity calculation of Block 4, which is $26M_{in}^2 + 6946816$, equation

is as below:

$$\begin{aligned} Space \sim O(\sum_{l=1}^{D} K_l^2 \cdot C_{(l-1)} \cdot C_l + \sum_{l=1}^{D} M^2 \cdot C_l) \\ = (3^2 \cdot 256 \cdot 512 + 3^2 \cdot 512 \cdot 512 + 3^2 \cdot 512 \cdot 512 + 2^2 \cdot 512 \cdot 512) \\ +((\frac{M_{in}}{8})^2 \cdot 512 + (\frac{M_{in}}{8})^2 \cdot 512 + (\frac{M_{in}}{8})^2 \cdot 512 + (\frac{M_{in}}{16})^2 \cdot 512) \end{aligned} \tag{4.23}$$

4.3.5 Block 5

input(224 × 224 RGB image)					
conv3-512	conv3-512	conv3-512	conv3-512	conv3-512	conv3-512
conv3-512	conv3-512	conv3-512	conv3-512	conv3-512	conv3-512
			conv1-512	**conv3-512**	conv3-512
					conv3-512
maxpool					

Table 4.6: Block 5 in VGG16

Block 5 has the same construct as Block 4, For the fifth maxpool layer, it will be:

$$W_{out} = \frac{W_{in} - W_{filter}}{S} + 1 = \frac{\frac{M_{in}}{16} - 2}{2} + 1 = \frac{M_{in}}{32} \tag{4.24}$$

$$H_{out} = \frac{H_{in} - H_{filter}}{S} + 1 == \frac{\frac{N_{in}}{16} - 2}{2} + 1 = \frac{N_{in}}{32} \tag{4.25}$$

So the feature dimension of the output feature map in Block 5 is $\frac{M_{in}}{32} \times \frac{N_{in}}{32} \times 512$. For the space complexity calculation of Block 5, which is $\frac{13}{2}M_{in}^2 + 8126464$, equation is as below:

$$\begin{aligned} Space \sim O(\sum_{l=1}^{D} K_l^2 \cdot C_{(l-1)} \cdot C_l + \sum_{l=1}^{D} M^2 \cdot C_l) \\ = (3^2 \cdot 512 \cdot 512 + 3^2 \cdot 512 \cdot 512 + 3^2 \cdot 512 \cdot 512 + 2^2 \cdot 512 \cdot 512) \\ +((\frac{M_{in}}{16})^2 \cdot 512 + (\frac{M_{in}}{16})^2 \cdot 512 + (\frac{M_{in}}{16})^2 \cdot 512 + (\frac{M_{in}}{32})^2 \cdot 512) \end{aligned} \tag{4.26}$$

4.3.6 Block 6

The first layer of Block 6 is a full connected layer, the input image dimension is $\frac{M_{in}}{32} \times \frac{N_{in}}{32} \times 512$, changing to 1×25088. The output dimension will be 1×4096. For

input(224 × 224 RGB image)
FC-4096
FC-4096
FC-1000
soft-max

Table 4.7: Block 6 in VGG16

second full conntected layer, the output dimension will be 1×4096. For the final full connected layer the dimension changed to 1×1000, which also is the dimenson of final output. For the space complexity calculation of Block 6, which is $4M_{in}^2 + 123638760$, equation is as below:

$$\begin{aligned} Space \sim O(\sum_{l=1}^{D} K_l^2 \cdot C_{(l-1)} \cdot C_l + \sum_{l=1}^{D} M^2 \cdot C_l) \\ = (1^2 \cdot 25088 \cdot 4096 + 1^2 \cdot 4096 \cdot 4096 + 1^2 \cdot 4096 \cdot 1000 + \\ + ((\frac{M_{in}}{32})^2 \cdot 4096 + 1 \cdot 4096 + 1 \cdot 1000)) \end{aligned} \tag{4.27}$$

So for the entire VGG 16 network, the space complexity can be approximately equal to

$$Space \sim O_{VGG16} \approx 304M_{in}^2 + \frac{1}{2}M_{in}^2 + 10^8 \tag{4.28}$$

Because the most important step in calculating the vc dimension is to calculate the space complexity of the model, through the above binary equation, it also need consider its layer numbers,finally shows that:

Definition 7.

$$VCD_{VGG16} = 304M_{in}^2 + \frac{1}{2}M_{in}^2 + 10^8 + 2 + 16$$

4.4 VC dimension of VGG 19

In fact, VGG 16 and VGG 19 are roughly the same, and it will be easier to calculate the space complexity of VGG 19 under the condition of controlling the input data.

4.4.1 Differences between VGG 16 and VGG 19

On the basis of VGG16, VGG 19 adds a convolution layer before the maxpooling layer of Block 3, 4, and 5, which makes three more layers in the final calculation of

the overall space complexity. So:

$$\begin{aligned} Space \sim O_{difference} &= (\sum_{l=1}^{D} K_l^2 \cdot C_{(l-1)} \cdot C_l + \sum_{l=1}^{D} M^2 \cdot C_l) \\ &= (3^2 \cdot 256 \cdot 256 + 3^2 \cdot 512 \cdot 512 + 3^2 \cdot 512 \cdot 512 + \\ &+ (\frac{M_{in}}{4})^2 \cdot 256 + (\frac{M_{in}}{8})^2 \cdot 512 + (\frac{M_{in}}{16})^2 \cdot 512)) \\ &= 26M_{in}^2 + 5308416 \end{aligned} \tag{4.29}$$

Which means, for the entire VGG 19 network, the space complexity can be approximately equal to

$$Space \sim O_{VGG19} \approx 330M_{in}^2 + \frac{1}{2}M_{in}^2 + 10^8 \tag{4.30}$$

So the VC-dimension will be:

Definition 8.

$$VCD_{VGG19} = 330M_{in}^2 + \frac{1}{2}M_{in}^2 + 10^8 + 2 + 19$$

Upon thorough analysis, it can be concluded that under identical conditions such as input data and activation function, VGG19 surpasses VGG16 in terms of learning capacity owing to the incorporation of three additional hidden layers. However, it is noteworthy that merely increasing the number of hidden layers may not be an optimal approach to efficiently harnessing the potential of deep learning within the VGG network. Based on the calculation of the VC dimension, it can be inferred that the addition of an extra fully connected layer or more blocks might prove more effective in this regard, as compared to adding one convolutional layer per block.

4.5 Conclusion

In conclusion, the VGG network has been a well-established neural network in the deep learning field, winning several ImageNet competitions and being widely used in various applications. Its unique network structure, with a stack of convolutional layers followed by a few fully connected layers, has shown great potential in image classification and object detection tasks.

Through the analysis of the VC dimension, it has been demonstrated that when the activation function, input data, and other variables are the same, VGG19 has a larger limit of learning ability than VGG16 due to the addition of three hidden

layers. This result supports the claim that the VGG19 network is more efficient in processing high-density data than VGG16, making it a better choice for complex tasks that require deeper and more complex neural networks.

Furthermore, this section aims to optimize the VGG neural network at the theoretical level. The mathematical reasoning provided in this study could help in developing more effective and efficient neural networks, particularly those that are capable of handling high-density data. For instance, the proposed optimization method through adding more fully connected layers or blocks could be applied to improve the performance of the VGG network or other deep learning models.

It is worth noting that the VCD approach used in this study is a powerful tool for analyzing and optimizing deep learning models. By analyzing the VC dimension of a neural network, researchers can gain insights into its learning ability and generalization performance. This information can be used to guide the design and optimization of neural networks for better performance and efficiency.

Overall, this study provides valuable insights into the VGG network and its optimization through the VCD approach. The findings demonstrate that the addition of hidden layers can significantly improve the learning ability of deep neural networks, but it is not necessarily the most efficient way to optimize them. Instead, adding more fully connected layers or blocks may be a better choice for achieving better performance and efficiency.

In the future, researchers can apply the VCD approach to explore and optimize other deep learning models, which may lead to new breakthroughs in various applications. With the rapid development of deep learning and artificial intelligence, the optimization of neural networks is becoming increasingly important, and the VCD approach offers a promising direction for achieving better performance and efficiency in deep learning models.

Chapter 5

Conclusion & Future Work

5.1 Overview

This thesis has presented various approaches to address the problem of VC dimension in different neural network. Although the results have shown good overall accuracy, there is still room for improvement in order to achieve optimal performance.

The contributions of this thesis can be valuable in the context of future work when researchers want to solve similar problems related to VCD on different neural network. By applying the methods in this, researchers and practitioners can improve the efficiency and accuracy of deep learning algorithms in a wide range.

Moreover, the findings of this thesis provide insights of deep learning, specifically in terms of VCD in the design and optimization of a new neural network. By deepening understanding of the mathematical properties of these models, we can further advance the field of deep learning and unlock new opportunities for innovation.

In summary, the approaches proposed in this thesis have demonstrated their potential to enhance the accuracy and efficiency of neural network by addressing the issue of VC dimension. These contributions can be used as a basis tool for future

research, and pave the way for the development of more effective deep learning algorithms.

5.2 Main Contributions

The main focus of this thesis was to provide theoretical support for the utilization of VC dimension (VCD) in enhancing neural networks. Through the mathematical calculation method, we were able to identify similar optimization techniques for other neural networks and provide distinct recommendations for improvement. The aim of this research was to not only enhance the effectiveness of neural networks through VCD, but also to provide a framework for future studies on optimizing neural networks. There are three key contributions of this thesis:

- **First** A statistical measure known as VCD was introduced into a single neural network, and its feasibility was demonstrated in this research. This not only rationalizes and explains the use of VCD in neural networks, but also provides ideas and methods for applying it to other types of neural networks.
- **Second** The feasibility of applying VCD for optimization in Autoencoder has been demonstrated in this study. Building on the previous application of VCD for a single neural network, this approach holds significant potential for enhancing the processing of Autoencoder, a neural network that serves both encoding and decoding functions.
- **Third** Through the application of VCD to the VGG neural network, we investigated the potential for achieving more accurate results through structural modifications. Our analysis also revealed that VGG19, under the theoretical framework of VCD, is more efficient in processing high-density data compared to VGG16. These findings provide theoretical support for improving the performance of the VGG network and shed light on the potential benefits of incorporating VCD into the optimization of other neural networks.

5.3 Conclusion

In this thesis, the concept of VC dimension was introduced and its relevance to neural network optimization was explored. VC dimension is a statistical term that involves

various mathematical calculations, and it was important to identify the most useful ones for neural network optimization. Uncertainties in neural networks such as the number of neurons, layers, and selection of activation functions needed to be taken into account during this process.

Two examples of neural networks were analyzed in the thesis, namely autoencoders and the VGG series networks. For autoencoders, the structure was broken down into encoder and decoder, making it easier to calculate the VC dimension. It was concluded that a more effective activation function can optimize the results of the autoencoder at the mathematical level.

For VGG networks, it was hypothesized that changing the number of neural network layers would be a more effective way to improve efficiency. Through mathematical reasoning and calculations based on VC dimension, it was shown that VGG19 was more efficient in processing high-density data than VGG16.

The thesis also emphasized the importance of considering uncertainties and complexities in neural networks during the optimization process. It is essential to take into account factors such as the number of neurons and layers, as well as the selection of activation functions, to achieve more accurate results.

In conclusion, this thesis provides a comprehensive exploration of the application of VC dimension to neural network optimization. It highlights the potential of this statistical method in improving the performance and efficiency of neural networks, and provides useful insights and recommendations for future research in this field.

5.4 Future Work

The future of neural network optimization through the lens of Vapnik-Chervonenkis Dimension (VCD) is promising. The concept of VCD has been introduced and its feasibility has been demonstrated in both single neural networks and autoencoders. Furthermore, it has been shown that VCD can also be used to optimize the widely-accepted VGG network, specifically through changes in the number of layers.

Moving forward, further research and exploration into the application of VCD to other neural networks can lead to further optimization and improvements in their performance. By considering uncertainties in neural network design, such as the number of neurons and selection of activation functions, more effective optimization strategies can be developed.

Additionally, the utilization of VCD in neural network optimization can also pro-

vide theoretical support for the practical design and development of neural networks. It can aid to identify more effective network structures and create more accurate and efficient models for different goals.

Overall, the future of neural network optimization through VCD is promising, and continued research and development in this area can lead to significant improvements in the performance and practical application of neural networks.

Bibliography

[1] Vapnik, V. (1998). Statistical learning theory. Wiley.

[2] Shalev-Shwartz, S., Ben-David, S. (2014). Understanding Machine Learning: From Theory to Algorithms. Cambridge University Press.

[3] Hastie, T., Tibshirani, R., Friedman, J. H., Friedman, J. H. (2009). The elements of statistical learning: data mining, inference, and prediction (Vol. 2, pp. 1-758). New York: springer.

[4] Mohri, M., Rostamizadeh, A., Talwalkar, A. (2018). Foundations of machine learning. MIT press.

[5] Goodfellow, I., Bengio, Y., Courville, A. (2016). Deep learning. MIT press.

[6] Hinton, G. E., Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. science, 313(5786), 504-507.

[7] Kingma, D. P., Welling, M. (2013). Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114.

[8] Vincent, P., Larochelle, H., Bengio, Y., Manzagol, P. A. (2008). Extracting and composing robust features with denoising autoencoders. In Proceedings of the 25th international conference on Machine learning (pp. 1096-1103).

[9] Simonyan, K., Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In Proceedings of the International Conference on Learning Representations (ICLR).

[10] Valiant, L. G. (1984). A theory of the learnable. Communications of the ACM, 27(11), 1134-1142.

[11] Bartlett, P. L., Shawe-Taylor, J. (1998). Generalization performance of support vector machines and other pattern classifiers. Advances in neural information processing systems, 10, 43-50.

[12] Kawaguchi, K. (2017). Generalization in deep learning. arXiv preprint arXiv:1710.05468.

[13] Zhang, C., Bengio, S., Hardt, M., Recht, B., Vinyals, O. (2016). Understanding deep learning requires rethinking generalization. arX

[14] Alpaydin, E. (2020). Introduction to machine learning (3rd ed.). MIT Press.

[15] Bishop, C. M. (2006). Pattern recognition and machine learning (1st ed.). Springer.

[16] Domingos, P. (2015). The master algorithm: How the quest for the ultimate learning machine will remake our world. Basic Books.

[17] Jordan, M. I., Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. Science, 349(6245), 255-260.

[18] Murphy, K. P. (2012). Machine learning: A probabilistic perspective (1st ed.). MIT Press.

[19] Jarrett, D., Stride, E., Vallis, K., Gooding, M. J. (2019). Applications and limitations of machine learning in radiation oncology. The British journal of radiology, 92(1100), 20190001.

[20] Bourilkov, D. (2019). Machine and deep learning applications in particle physics. International Journal of Modern Physics A, 34(35), 1930019.

[21] Du, L., Ma, N., Dai, X., Yu, W., Huang, X., Xu, S., Liu, F., He, Q., Liu, Y., Wang, Q., Liu, X., Zheng, H., Qu, B. (2020). Precise prediction of the radiation pneumonitis in lung cancer: An explorative preliminary mathematical model using genotype information. Journal of Cancer, 11(8), 2329-2338. doi: 10.7150/jca.37708. PMID: 32127959; PMCID: PMC7052914.

[22] Xiao, L. S., Li, P., Sun, F., Zhang, Y., Xu, C., Zhu, H., ... Zhu, H. (2020). Development and validation of a deep learning-based model using computed tomography imaging for predicting disease severity of coronavirus disease 2019. Frontiers in bioengineering and biotechnology, 8, 898.

[23] Wu, Y., Giger, M. L., Doi, K., Vyborny, C. J., Schmidt, R. A., Metz, C. E. (1993). Artificial neural networks in mammography: application to decision making in the diagnosis of breast cancer. Radiology, 187(1), 81-87.8

[24] Ravì, D., Wong, C., Deligianni, F., Berthelot, M., Andreu-Perez, J., Lo, B., Yang, G. Z. (2017). Deep learning for health informatics. IEEE journal of biomedical and health informatics, 21(1), 4-21. https://doi.org/10.1109/JBHI.2016.2638018

[25] Esteva, A., Kuprel, B., Novoa, R. A., Ko, J., Swetter, S. M., Blau, H. M., Thrun, S. (2017). Dermatologist-level classification of skin cancer with deep neural networks. Nature, 542(7639), 115-118.

[26] Khajehnejad, M., Habibollahi Saatlou, F., Mohammadzade, H. (2017). Alzheimer's disease early diagnosis using manifold-based semi-supervised learning. Brain sciences, 7(8), 109.

[27] Sutskever, I., Vinyals, O., Le, Q. V. (2014). Sequence to sequence learning with neural networks. In Advances in neural information processing systems (pp. 3104-3112).

[28] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention is all you need. In Advances in neural information processing systems (pp. 5998-6008).

[29] Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L. D., Monfort, M., Muller, U., Zhang, J., Zhang, X., Zhao, J., Zieba, K. (2016). End to end learning for self-driving cars. arXiv preprint arXiv:1604.07316.

[30] Google. (2016). GoogleNet: Going deeper with convolutions. https://ai.googleblog.com/2016/05/googlenet-inception-v1-for-image.html

[31] Bansal, T., Krone, M., Huang, J. (2018). Social LSTM: Human trajectory prediction in crowded spaces. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 9612-9620. doi: 10.1109/CVPR.2018.01006

[32] Bengio, Y., LeCun, Y. (2007). Scaling learning algorithms towards AI. Large-scale kernel machines, 34(5), 1-41.

[33] Sontag, E. D. (1998). VC dimension of neural networks. NATO ASI Series F Computer and Systems Sciences, 168, 69-96. Springer Verlag.

[34] Kárný, M., Warwick, K., Kůrková, V. (1998). Recurrent neural networks: Some systems-theoretic aspects. In Dealing with Complexity (pp. 1-12). Springer.

[35] Lawrence, S., Giles, C. L., Tsoi, A. C. (1997). Lessons in neural network training: Overfitting may be harder than expected. In AAAI/IAAI (pp. 540-545). Citeseer.

[36] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research, 15(1), 1929-1958. JMLR. org.

[37] Hoeffding, W. (1994). Probability inequalities for sums of bounded random variables. In The Collected Works of Wassily Hoeffding (pp. 409-426). Springer.

[38] Chen, J. (2008). Research and Application of Support Vector Machine Regression Algorithm. (Master's Thesis). Jiangnan University. Wuxi, China.

[39] Abu-Mostafa, Y. S. (1989). The Vapnik-Chervonenkis dimension: Information versus complexity in learning. Neural Computation, 1(3), 312-317. MIT Press One Rogers Street, Cambridge, MA 02142-1209, USA journals-info ....

[40] Vapnik, V. N., Chervonenkis, A. Ya. (2015). On the uniform convergence of relative frequencies of events to their probabilities. In Measures of Complexity (pp. 11-30). Springer.

[41] McCulloch, W. S., Pitts, W. (1956). A logical calculus of ideas immanent in nervous activity. Avtomaty [Automated Devices] Moscow, Inostr. Lit. publ, 363-384.

[42] Blumer, A., Ehrenfeucht, A., Haussler, D., Warmuth, M. K. (1989). Learnability and the Vapnik-Chervonenkis dimension. Journal of the ACM (JACM), 36(4), 929-965.

[43] Bartlett, P. L., Harvey, N., Liaw, C., Mehrabian, A. (2019). Nearly-tight VC-dimension and pseudodimension bounds for piecewise linear neural networks. The Journal of Machine Learning Research, 20(1), 2285-2301. JMLR. org.

[44] Pinto, L., Gopalan, S., Balasubramaniam, P. (2021). On the stability and generalization of neural networks with VC dimension and fuzzy feature encoders. Journal of the Franklin Institute, 358(16), 8786-8810. Elsevier.

[45] Paul, A. N., Yan, P., Yang, Y., Zhang, H., Du, S., Wu, Q. M. (2021). Non-iterative online sequential learning strategy for autoencoder and classifier. Neural Computing and Applications, 33(23), 16345-16361. doi: 10.1007/s00521-021-06287-2

[46] Zhou, Z. (2016). Machine learning [M]. Tsinghua University Press.

[47] Shalef-Shwartz, S., Ben-David, S. (2016). Deep understanding of machine learning: From principle to algorithm [M]. Machinery Industry Press.

[48] Abumostafa, Y. S., Magdon-Ismail, M., Lin, H. T. (2012). Learning from Data: A Short Course [J]. AMLBook.

[49] Han, J., Moraga, C. (1995). The influence of the sigmoid function parameters on the speed of backpropagation learning. In International Workshop on Artificial Neural Networks (pp. 195-201). Springer.

[50] Weisstein, E. W. (2003). Hyperbolic Functions. MathWorld. Wolfram Research, Inc. https://mathworld.wolfram.com/HyperbolicFunctions.html

[51] Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., LeCun, Y. (2013). Overfeat: Integrated recognition, localization and detection using convolutional networks. arXiv preprint arXiv:1312.6229.

[52] Liu, H. (2022, November 04). VC Dimension. In Encyclopedia. https://encyclopedia.pub/entry/32803

[53] Grigorescu, S., Trasnea, B., Cocias, T., Macesanu, G. (2020). A survey of deep learning techniques for autonomous driving. Journal of Field Robotics, 37(3), 362-386.

[54] Wikipedia. (2021). VAE Basic [PNG]. Retrieved from https://en.wikipedia.org/wiki/File:VAE$_B$$asic.pngKumar, V., Nandi, G.C., Kala, R.(2014).Static$
104).doi : 10.1109/IC3.2014.6897155.

[55][56] Paul, A. N., Yan, P., Yang, Y., Zhang, H., Du, S., Wu, Q. J. (2021). Non-iterative online sequential learning strategy for autoencoder and classifier. Neural Computing and Applications, 33(23), 16345-16361.

[57] Karimpouli, S., Tahmasebi, P. (2020). Physics informed machine learning: Seismic wave equation. Geoscience Frontiers, 11(6), 1993-2001.