An Intelligent System Conception For Clinical Decision Support

Francis Anokye (francis.anokye@aims-senegal.org)
African Institute for Mathematical Sciences (AIMS)
Senegal

Supervised by: Prof. Seydina Moussa Ndiaye
SeySoo Sarl & Université Virtuelle du Sénégal, Senegal
Co-Supervised by: Dr. Cheik Birahim Ndao
African Institute for Mathematical Sciences, Senegal

February 2019
Submitted in Partial Fulfillment of a Masters II at AIMS

AIMS | African Institute for Mathematical Sciences
SENEGAL

Abstract

The concept of clinical decision support has undoubtedly received massive world-wide attention and it is utilized in the healthcare domain to assist with complex treatment and diagnostic decisions. Even though some healthcare facilities in Senegal have been quite responsive to this modern developments to support medical knowledge and decision tasks, it is noted that these systems are not intelligent. The growing size of biomedical data and the complex advancement in the intensive techniques that are required to accurately associate a disease with its radiology medical image such as computerized tomography (CT) scan, magnetic resonance imaging (MRI), X-ray image etc, and treatment based on specific biomarkers by radiologists and clinicians, greatly impact on the outcome of medical diagnostic decisions. The consequences of misinterpretating medical images as a result of wrong association (classification) of their modalities with their respective parts and diseases can not be over-emphasized. They present huge vacua in the outcome of medical diagnosis and contribute greatly to the chance of wrong diagnosis, which may lead to adverse life threatening effects on treatments. In this report, we draw on the intelligent conception of transfer learning from a fine-tuning-based deep learning approach as a distinguished tool for medical image classification. Experiments are conducted with radiology images on two state-of-the-art pre-trained networks, Inception-V3 and VGG16, from the imagenet competition to build intelligent patient models for radiology medical image classification which would be adopted for integration into an already existing medical software to efficiently provide realtime clinical decision support to clinicians in Senegal. Our results demonstrate the superiority of the VGG16 pre-trained model for boosting the classification performance on medical images.

Declaration

I, the undersigned, hereby declare that the work contained in this essay is my original work, and that any work done by others or by myself previously has been acknowledged and referenced accordingly.

[illegible]

Francis Anokye, 24 January 2019

Acknowledgements

The Almighty God has been merciful to me and I am forever grateful for his love and favour upon me. My greatest appreciation goes to the African Institute for Mathematical Sciences, Senegal, for introducing me to new knowledge of Science and Technology, which has helped me to answer many profound questions about my career interests and future aspirations through the academic works, scientific research and industrial experience. God bless the efforts of all the lecturers, tutors and the support of all sponsors who made my training a dream come true. It was humbling to benefit from the coaching of my supervisor, Dr. Seydina M. Ndiaye, and co-supervisor, Dr. Cheik Birahim Ndao, without whom this report wouldn't have been realized. A special mention is made for Dr. Samuel Ofori-Agyekum for assisting me with his medical knowledge. My ultimate thanks finally goes out to my family for their unending prayers and support, and to all who have shown me goodness and support in this great country, Senegal.

Contents

Abstract **i**

1 Introduction **1**

1.1 Background 1

1.2 Statement of the Problem 3

1.3 Objectives 3

1.4 Significance of the study 4

1.5 Motivation 4

1.6 Structure of report 5

2 Review of Literature **6**

2.1 Historical Developments 6

2.2 Reasoning Concepts of Inference 7

2.3 Machine Learning for decision support 9

3 The Intelligent Conception **12**

3.1 Machine Learning 12

3.2 Artificial Neural Networks (ANN) 14

3.3 Structure of Artificial Neural Networks 15

3.4 Cost Function 20

3.5 Optimizing The Cost Function 21

3.6 Regularization Techniques 25

3.7 Convolutional Neural Networks 28

3.8 Transfer Learning 31

3.9 Pre-Trained Models From ImageNet and ILSVRC 33

4 Experimental Results **39**

4.1 Setting 39

4.2 Hardware and Software Setup 39

4.3 The Data Preparation 40

4.4 VGG16 Implementation 41

4.5 Inception-V3 Network Implementation 42
4.6 Model Evaluation 42
4.7 Performance - VGG16 44
4.8 Performance - Inception-V3 46

5 Conclusion **49**

References **58**

1. Introduction

1.1 Background

Senegal like many other developing countries is not exempted from the league of countries that are severely faced with challenges of providing quality health care services to her populace, especially for many rural and semi-urban areas. The health system is comprised of the public and private sectors (including NGOs). The public sector has 35 hospitals, 99 health centers and 1,237 public health posts [17]. Among the several reasons contributing to the challenges faced by the health sector include limiting factors such as low rainfall as a result of climatic conditions, degradation of the natural resource base due to environmental challenges, and the lack of cross-border disease control [17]. These barriers to quality health care in Senegal can not as well be entirely blamed on the lack of adequate trained healthcare workers, and or social and religious barriers, but there also exist a great deficit on how to support the knowledge role of healthcare decision-makers in their decision tasks [16].

Prevention of diseases is the topmost priority on the national health agenda of Senegal, but treatment is the predominant reality. New kinds of innovation such as those that change the manner in which consumers use health care, use technology to improve health care and also implement business models that integrate separate health care activities [58] have since been adopted to improve healthcare delivery and treatment options. Healthcare providers are now taking advantage of technology and data to ease the complex process of their daily patient treatment practices. Among the recent approaches adapted to healthcare in Senegal is the use of low-cost innovations powered by mobile technology to improve health management information systems and contraceptive logistics, which assist health workers with information when they need to make decisions. [4].

1.1.1 Adoption of Decision Support System in Healthcare. The concept of decision support has undoubtedly received massive world-wide attention and it is utilized in the healthcare domain to assist in complex treatment and recommendations [30]. Decision support systems may be either fully computerized, human-powered or a combination of both. Healthcare is by far one of the most crucial and important components of every society's welfare and progress. The episodes of pressure mounted on healthcare systems to be more productive, create improvements in the delivery of services and further reduce cost [110] have been great contributors to the recent rapid adoption of technology-based systems in the healthcare sector globally. The virtual explosion of health information through the internet as a result of these adopted technology-based concepts is exerting massive positive impact on clinical decisions made by clinicians, staff and patients [30]. The vast availability of data has further given way to the creation of support systems that enhance decision-making activities.

The proposal by Rajalakshmi et al. [103] to implement decision support system in the healthcare was to integrate input patient data from relevant sources into an autonomous data warehouse, which would ensure that significant and adequate information (in the form of output presentation) are made available to the clinicians at the front-end. This idea has been phenomenal and has brought much positive effect in all aspects of the healthcare system where it is been implemented. Senegal has been quite responsive to this new evolution by adopting the use of mobile technology to assist health workers with information during periods of decision-making. Many healthcare delivery centers like hospitals have gained much competitive edge over their competitors, and both higher and lower levels of decision-making tasks within such environments do not suffer much in relation to time conservation which ensure that patients receive relatively good care in the shortest possible time.

Decision support system is a computer-based system that provides valuable assistance to organizations and businesses during periods of decision-making. Several industries are currently adopting decision support systems and it is predicted that 30 percent of health providers will use cognitive analytics with patient data by 2018 [10].

1.1.2 Clinical Decision Support System. Health Level Seven International (HL7) defines clinical decision support as an 'act' of providing clinicians, patients and other healthcare stakeholders with the knowledge or specific information intelligently filtered or presented at appropriate times, to enhance health care. For the purposes of this paper, Health IT's definition is adopted which describes clinical decision support as a 'system' that intelligently filters person-specific information at the appropriate time and provides medical staff with the knowledge required to enhance health care delivery (HealthIT.gov). There are three primary components that make up a typical decision support architecture: the knowledge-base database system, the patient model (the context inference engine based on patient data) and the graphical user interface (GUI) for user interactivity. Accurate data standards and structuring together with system interoperability that ensure scalable and maintainable clinical integration and deployment of services are the invisible pillars that cushion data-driven decision support systems to function accurately as desired.

Although, there exist several limiting factors such as uncertainty in medical knowledge, integration into workflow and organizational challenges that are influencing the adoption of CDSS [30], yet great results from research studies have also given evidences to butress the conviction that clinical decision support systems help reduce the errors in medical decision processes which further improve health care quality [34], [70], [116].

A clinical decision support system is intelligent when it is able to gain knowledge automatically from patient data and make predictive analysis to help users (clinicians and medical staff) arrive at decisons more accurately and faster while conserving time. Clinical decision support systems offer decision-making guidance to medical staff in relation to the relevant context of usage eg. during diagnosis, treatment decisions, appointments etc while making use of available patient data. It is to be noted that they can not be used as substitutes for medical staff [5]. They provide opinions during periods of medical diagnosis. The final decision always lies in the bosom of the clinician or the individual using the support system.

1.1.3 The Medical Diagnosis Process. General medical knowledge describes medical diagnosis as a process used to identify which disease best fits a patient's health condition, signs, and symptoms. Many diseases have similar signs and symptoms so it is very important to go through this process to identify the exact disease. Diagnosis is the most important part of the medical process because without a proper diagnosis it is impossible to receive the best treatment. The process begins with taking the patient's historical data, Anamnesis morbid (history of disease of both the patient and that of the family) and anamnesis vitae (history of patient's lifestyle). The signs and symptoms are then reviewed in lieu of the history. After that comes physical examination. The clinician (doctor) examines the patient physically for further clues to confirm or rule out the possible diseases the patient suffers or will be suffering from. The type of examination performed is decided based on the patient's signs and symptoms together with the history. The clinician collects the evidence and attempts to discover all the possible causes. At this stage, if the clinician is able to identify the exact disease, it is called a clinical diagnosis. It is of much importance to note that not all diseases can be diagnosed clinically. Some need further diagnostic investigations like laboratory test analysis, instrumental investigations like radiology X-ray images, MRI scan images, Ultrasound etc. In other cases, samples of the patient's tissues must be taken for biochemical and pathological analyses.

The above processes laden clinicians to be accurate with his or her decision based on the data of the patient in order to avoid consequential effects on the said patient. For this reason, the process of diagnosis is very challenging and obliges clinicians to possess a broad array of knowledge or information [124] about diseases, their best treatments, expected side effects, and etc during a patient interaction. In diagnoses, clinicians want to minimize the costs of medical tests and maximize the diagnostic power. This is an important goal as clinicians sort out patients accurately based on their needs and associated risks. Finally, clinicians can reduce the time and medical resources and diagnose correctly [30] while ensuring a reduction in prescription and medication error rates [69] by relying on the aid provided by intelligent clinical decision systems.

1.2 Statement of the Problem

Medical diagnosis of patient's health condition can be assessed in several ways ranging from or the combination of the patient's own description of symptoms, physical examination by the clinician, interpretation of medical images and laboratory tests results. After the diagnosis, treatment is prescribed having in mind the potential side-effects of the patient's condition to that treatment. Treatment effects are nearly impossible to eliminate; but they can often be reduced [50]. The global awareness created by the US Institute of Medicine through their report 'To Err Is Human' [41] emphasized that, medical error is a major cause of preventable mortality, morbidity and inappropriate use of resources.

The consequences of misinterpretating medical images can not be over-emphasized. They present huge vacua in the outcome of medical diagnosis and contribute much to the chance of wrong diagnosis which may lead to adverse life threatening effects from treatment. Due to the growing size of biomedical data and the complex advancement in the intensive techniques that are recently employed to associate a disease with its radiology image and treatment based on specific biomarkers, a clinician or radiologist's ability to differentiate between these biomedical images and their modalities greatly impact on the outcome of diagnostic decisions [74].

Even though some healthcare facilities in Senegal have been quite responsive to this modern developments to support medical knowledge and decision tasks, it is noted that these systems are not intelligent. They lack the ability to learn from new patient data, and this negates their efficiency and effectiveness in a setting where they are needed most to compensate for the extremely high patient-to-doctor ratio while ensuring that time and resources are conserved and patients receive the optimal care.

1.3 Objectives

The general goal of this study is to build an intelligent patient model for clinical decision support using the concept of Machine Learning for radiology medical image classification which aims at improving clinical performance through the provision of real-time diagnostic support.

The specific objectives are;

- to use the concept of transfer learning to build patient model from medical radiology image data generated from Medicis which would be integrated into an existing software to provide intelligent clinical decision support.

- and also to identify which of the pre-trained state-of-the-art models from Imagenet competitions offer the best predictive performance for our support system.

1.4 Significance of the study

The significance of this report is owed to the inevitable importance of healthcare in our everyday lives as opposed to the many challenges that have created a huge gap between optimal evidence-based medical practice and the actual care that is received in healthcare facilities in Senegal.

The main contribution of our work is expected to introduce the concept of transfer learning using fine-tuning techniques for deep neural networks to boost clinical decision performance in Senegal in order to alleviate the gap between medical knowledge, expensive financial resources and workload on clinicians and radiologists by distinguishing among the two state-of-the-art models, Inception-V3 [122] and VGG16 network [115] from the imagenet competition [109], the one that offers the best predictive ability through the effective transfer of knowledge from their respective computer vision for natural image classification tasks to the task of medical radiology image classification to function as the intelligent engine for clinical decision support.

The report further credits the satisfaction derived in the adoption of transfer learning as a leverage in situations of image classification tasks where the training data in a target domain are not sufficient to be trained from scratch as evidenced in many studies. It further proves the conviction associated with learning problems where the pre-trained data and new target data do not necessarily follow the same distribution.

1.5 Motivation

The major improvement in computer resources have brought about massive elimination of several limitations on how computer was formally used to provide assistance in medical care. This has further translated into a new development where human intelligence is gradually merging with the technology to create artificial intelligence (AI), and this is transforming how healthcare is delivered in many positives ways. A new concept of intelligence is attributed to CDSS as a result of their use of AI techniques to learn from medical data. The CDSS are basically designed by combining the knowledge-based reasoning techniques of AI and the basic architectural functionalities of DSS. They may be programmed with the electronic health record (EHR), health information system (HIS) or any other information system available in the healthcare facility in order to mine the vast amount of patient data and analyze the complex patterns which may sometimes not be visible to even medical specialists [12].

Machine learning is a proven particular approach to AI which has specially introduced deep learning from which deep convolutional neural networks (CNN) is being applied in the areas of computer vison to build image classifiers with optimized and accurate performance. This is transforming the healthcare industry by improving diagnostic results; gradually helping to relief the burden of over-reliance on the few medical staff (especially radiologists and doctors) for all medical diagnostic activities while improving the precision of medical image classification and interpretation at the expense of the errors to the minimal levels.

Deep convolutional neural network architecture that was trained on huge amount of image data to extract meaningful patterns for classification task could be transfered to similar task of medical radiology image classification. This intelligent concept called Transfer Learning which is adopted in the design of our of patient model outputs results that help clinicians and radiologists to classify medical images more accurately during medical diagnostic activities.

1.6 Structure of report

In this chapter, I have

- introduced the big picture of the problem and context in which the study is based.
- provided brief background information about clinical decision support systems.
- introduced the need for the adoption of machine learning in clinical decision support.
- indicated what I set out to achieve in this study, and how.

The remaining chapters are organised as follows:

Chapter 2 provides a review of the historic development of clinical decision support and some related works in which intelligent concepts from machine learning had been implemented to offer medical decision support.

Chapter 3 describes the fundamental and theoretical basics of our methodology that is adopted in this work .

Chapter 4 presents the findings from our adopted methodology.

Chapter 5 describes the conclusion and potential directions of our work as a result of the critical analysis of tools used in the current work.

2. Review of Literature

This chapter provides a brief review of the historic development of clinical decision support. Next, we consider relevant reasoning concepts of inference used in clinical decision support design and finally consider related works in which intelligent concepts from machine learning had been implemented to offer decision support.

2.1 Historical Developments

After the second world war, Alan Turing approached the key fundamental question of artificial intelligence (AI) by combining his concept of the universal computer [126] and practical experience in building code-breaking systems. He defined the intelligent behavior of a computer as its ability to achieve human-level performance in a cognitive task [94]. McCulloch and Walter Pitts proposed a model of artificial neural networks (ANNs) to model the human brain, in which each neuron was postulated as being in the binary state [88], and that the simple networks which were made by connecting neurons could learn. This model influenced John von Neumann to support and encourage his two graduate students which eventually led to the invention of the first neural network computer in 1951. Three major branches of machine learning emerged after these periods. Classical work in symbolic learning is described by Hunt et al. [66], in statistical methods by Nilsson [96] and in neural networks by Rosenblatt [108].

After this period was followed by historical cycles of disillusionment [36] and funding cuts by governments as a result of the overhyped potentials of AI to resolve specific domain problems. Most of the AI programs developed by researchers could not meet the numerous optimistic expectations of solving real-world problems. This was because most of the programs lacked knowledge about the problem domain and attempted problems were too broad which attracted polynomial time to arrive at conclusions.

However, the development of the first expert systems such as DENDRAL [24], MYCIN [113] and PROSPECTOR [53] in the 1970's led to the maturity of expert system technology and its massive applications in different areas in the 1980's and 90's. This new development of expert systems created knowledge engineering, the process of building intelligent systems. This was achieved through the restriction of the general problem domain weak methods to specific domains inspired by knowledge-intensive methods. This enabled the computers to perform at human-expert level. This achievement promoted the emergence of Fuzzy expert systems [25] which model humans' sense of words, decision making and common sense based on Fuzzy logic and knowledge-based expert systems such as rule-based expert systems [56]. Since then, AI has grown beyond the bounds of mere curiosity to a valuable tool which supports humans in making decisions.

In 1986, Waterman [131] revealed through his survey that, out of nearly 200 expert systems which were successfully been applied in different application areas, most of these applications were in the field of medical diagnosis. But, the tables had turned when a similar survey was conducted by Durkin [44] in 1994. The survey proved that the areas of business and manufacturing had taken over the use of expert systems to improve business performance. The major problems that surrounded the use of expert systems were that they could neither learn nor improve themselves through experience. They demanded so much efforts for their development [94].

The development of minicomputers and operating systems gave computer tasks and processes ample time to execute and run without problems, hence, giving way to the practical computerization of decision support systems [102]. This inspired the emergence of several Machine Learning (ML) algorithms and eventually led to the rebirth of artificial neural networks (ANN's) in the 1980's. This new development made it possible for ML algorithms to learn from historical data automatically and improve the adaptability, error tolerance and the speed of knowledge-based systems to provide decision support.

Earlier versions of DSS's existed in the form of text documents and flowcharts that lacked scientific evidence to guide practice. They provided guidelines to their users but lacked the ability to interact with them. However, the discoveries that surrounded the studies conducted between the 1950's and 1980's by Carnegie Institute of Technology (CIT) and Massachusetts Institute of Technology (MIT) led to the adoption of numerous technologies in the design of DSS and their further usage in the healthcare [132]. As a result of these studies, new platforms were created from the several algorithms that were developed. They were to offer scientific and evidence-based guidance by way of modeling and analyzing the large sets of data that were generated in the healthcare domain. Prior to the mid-1970's, there was little need for evidence to support the design of clinical decision support systems to help with clinical decisions since the practice was largely controlled by clinicians who had adapted themselves to several years of intuition and subjected the practice to their own experiences [111].

In the work of Sittig et al. [116], Fox et al. [48] and Velickovski et al. [129], they draw on the benefits of CDSS and highlight some setbacks by reinforcing that the best practices in clinical decision support design, their development, and implementation, creation of their architecture for sharing modules and services and creating internet-accessible repositories are the most common crucial challenges encountered during the design, implementation, and deployment of CDSS in healthcare. However, Berner et. al. [21] predicted that although several of these challenges abound and surround the integration of those newly created platforms, yet indications from the increased support for clinical computing, changes in government policies and the general increase in computer literacy rates were great potentials that could resolve the challenges.

2.2 Reasoning Concepts of Inference

The modern paradigm of CDSS usage for decision support simply means that the clinician interacts with the CDSS, makes use of his or her own knowledge and that of the output from the CDSS to make better inferences from the patient's data. Typically, a CDSS makes suggestions for the clinician to look through, and the clinician is expected to pick out useful information from the presented results and discount erroneous suggestions [50]. Clinical decision support systems are classified based on the reasoning concepts they adopt for their inference which emanates from the specific task they are given to execute.

The reasoning engine justifies the recommendations or decisions output by the CDSS. Velickovski et al. [129], in their design of CDSS for the early detection and assessment of chronic obstructive pulmonary disease (COPD), made their architectural model selection from four principal models. They settled for the service-oriented model due to its unique coverage of features which combined the functionalities of sharing services across clinical centers, reducing the manual entry of patient data into CDSS due to its connection or dependence on electronic health record (EHR). The stand-alone model, integrated model, and standards-based model were the other architectural models they considered. All communication within COPD and between its external systems during decision support activities are coordinated by the CDSS controller which manages the running of the reasoning engine. See [133] for an extensive review of these architectural models.

2.2.1 Rule-Based Reasoning. In the early days, the rule-based reasoning was adopted by most of the CDSSs. Hence, they were conceived as being used to make decisions for the clinician to just follow. The clinician would input patient data and wait for the CDSS to output the right decision and the clinician would simply act on that output. Rule-based systems [57] match their input against a set of IF-THEN conditions and, if some of these match, a rule (executed as mathematical knowledge in the form of logic) corresponding to one of the conditions is chosen, and the action associated with the rule is executed. The Rete algorithm [46] functions as an efficient pattern matching algorithm for implementing rule-based systems, hence, it is mostly used as the inference engine for rule-based reasoning.

INTERNIST [90], MYCIN [113], [112] and ONCOCIN [59] are examples of rule-based systems that were earlier built to support clinical decision tasks. They capture medical knowledge by linking rules together until a final decision is attained [129]. INTERNIST was designed at the University of Pittsburgh in 1974 for the diagnosis of complex diagnosis of complex problems in general internal medicine. MYCIN was designed consisting of about 450 independent rules of IF-THEN conditions derived from human knowledge through extensive interviews of experts to diagnose bacteria causing severe infections, and to recommend treatment (antibiotics) for certain blood infections (antimicrobial selection for patients with bacteremia or meningitis). It could perform at a level equivalent to human experts by providing therapeutic advice in a user-friendly and convenient manner. ONCOCIN was designed at Stanford University to assist physicians with the treatment of cancer patients receiving chemotherapy [59].

2.2.2 Guideline-based Reasoning. In the work of Chih-Lin [30], he describes four important aspects that need to be considered in the process of developing a guideline-based decision support system [38]. The first is the construction of a guideline model which focuses on the representation of the guidelines. The second is the knowledge acquisition guideline to facilitate the process of acquiring knowledge directly from a domain expert. The last but one step is the verification and testing of the guidelines using existing patient data to ensure that the guidelines are syntactically and semantically correct and free from ambiguity. Finally, is the guideline execution, which focuses on the execution time and ensures the guideline engine can run in multiple clinical domains and in various modes. This approach is practice-based and knowledge exists in the form of guidelines. However, this conventional practice-based approach is prone to several difficulties.

Fox et al. [50], outlined their succinct convictions about the massive contribution that could be achieved when evidence-based clinical practice is implemented as a new strategy to improve the manner in which guideline-based and practice-based clinical practices are implemented. They believe that the problems that are frequently encountered during the use of conventional clinical practice guidelines (CPG) could be allayed to some level of degree when medical knowledge and decision logic are formally presented in formats that can be applied and interpreted by computers to offer decision support services. CPG could be a text or electronic document that include clinical statements, algorithms or flowcharts that direct and recommend [47] with reasons and evidence how clinicians are to practice in relation to specific conditions. This method is evidence-based and relies upon a more explicit and logical approach that capture information in a more human-friendly manner and form.

The use of guidelines with logical flows to manipulate and make an inference from clinical data which are mostly executed with control structures to arrive at decisions through the entire workflow procedure [47] functions as the reasoning concept to support clinical judgment and decision-making. PROforma [49], Arden Syntax [35] and GuideLine Interchange Format [98] are examples of guideline-based reasoning support systems. [1] summarizes the processes used by the American Academy of Family Physicians (AAFP) to produce evidence-based guidelines.

2.2.3 Probabilistic Reasoning. Clinical decision support systems rely on statistical techniques to acquire clinical knowledge through the combination of explicit models and knowledge derived from historical patient data to form the basis of their decision. Ledley and Lusted [80] in their paper "Reasoning Foundations of Medical Diagnosis" lay out how logic and probabilistic reasoning help in diagnosis and treatment selection in medicine. Probabilistic reasoning naturally presents itself in almost all medical diagnostic processes due to the inevitable elimination of uncertainties in diagnostic outcomes. It is problematic to assume that all medical decisions are 'absolutely' true and guaranteed to occur as predicted. Probabilistic reasoning becomes useful when the system under consideration is inherently prone to non-deterministic randomness or when we lack the full features or variables to accurately tell the behavior of that system.

Bayesian networks [100] are used as the concise representation of probabilistic knowledge. Himes et al. [60] use Bayesian networks and graphical representation to describe the probabilistic relationships between diseases and symptoms with conditional probabilities. They constructed a Bayesian network to identify the most probable network of dependency from a dataset comprising 9,349 patients from which 843 were cases and 8,506 were controls. They explored the most probable model against different networks which were scored by their posterior probabilities, and the one with the maximum posterior probability was returned using 109 clinical variables with the K2 algorithm. The network that contained all the 109 variables had several associations among the variables, but several of these relationships had no direct influence on COPD.

The notion of probability provides means of quantifying the uncertainty associated with a decision output by a system. Judea Pearl [101] gives an orderly illumination of probability as a language for reasoning in her book 'Probabilistic Reasoning In Intelligent Systems'. It, however, describes the attitude of AI researchers as 'adamant' towards the use of probability because it is practically impossible to list all possibilities due to the poor nature of people as bad estimators of probability and how inconvenient they find the use of it.

2.3 Machine Learning for decision support

Machine Learning (ML) has been widely used in diagnostic and health care applications as the reasoning engine of inference. In [52], Garg et al. hinted at how researchers have suggested the application of machine learning (data mining) techniques as the most suitable options in clinical support systems for diagnosis. This is because ML provides tools for dealing with the difficulties of incompleteness or missing parameter values, random noise in the data, sparseness ie. few and or non-representable patient records available, and inappropriate selection of parameters for a given task which are often encountered when learning from patient data [85]. Numerous techniques from ML have been widely used in clinical applications for disease diagnosis which improve patient care, reduce investigation time and errors and also improve the performance of medical practitioners. For example, techniques such as decision trees, ANN's, Bayesian networks, support vector machines (SVM), kernel density, bagging algorithm have been actively used in clinical support systems for the diagnosis of heart disease. [37], [118], [118].

According to the work of Safdar et al. [65] in which they reviewed ML based decision support systems (DSS) for heart disease diagnosis, 331 studies were reviewed out of which 20 met their inclusion criteria. They discovered that most of the studies related ischemic heart diseases with the ANN as the commonest ML technique implemented as the reasoning engine for inference. Closely related is the report of Lisboa et al. [82] which provides a systematic review of how ML algorithms such as ANN's and SVM gain knowledge automatically from data and offer decision support in cancer diagnosis. They provide an assessment of the benefits of ANN's as decision-making tools in the field of cancer. Moustakis and Charissis [93] also surveyed the role of ML in medical decision-making and provide an extensive literature review on various ML applications in medicine that could be useful to practitioners to improve the efficiency and quality of decision making systems in medical applications. ML techniques have been applied to a variety of medical domains to improve medical decision-making [72]. Practical guidelines on how to select the most suitable algorithm for a specific medical application is found in [77].

Supervised learning systems for classification have been successfully applied in a number of medical domains, for example, in the localization of a primary tumor, prognostics of recurrence of breast cancer, diagnosis of thyroid diseases, and rheumatology [107]. The SVM [128], which was originally designed for binary classification problems [18], is one of the most actively developed classifiers in the ML community, haven been successfully applied to a number of medical problems [86], [104]. Dreiseitl et al. [42] compare five classification algorithms for the diagnosis of pigmented skin lesions. Their results show that logistic regression, ANN's, and SVM perform comparably, while k-nearest neighbors (KNN) and decision trees perform worse [26], [27], [33]]. Acir and Guzelis [13] also applied SVM in automatic spike signal detection in electro encephalo grams (EEG), which is used in diagnosing neurological disorders related to epilepsy. ANN is applied in [68] to classify lung sound signals into six different categories to assist diagnosis.

ML based CDSS was develped in the work of Oguntimilehin et al. [97] for the diagnosis and treatment of typhoid fever in Africa. The system improves upon some earlier intelligent systems [14], [125] that were developed for the diagnosis of malaria and typhoid fever. In this system, a decision tree classifier was built to function as the inference engine for decision-making which solves the problems of diagnosis without treatment, lack of implementation, and ineffective evaluation of system performance. However, the generalization ability of the reasoning decision tree classifier model on future patient data for accurate diagnosis is prone to the danger of inaccurate predictions which could negatively impact the outcome of diagnostic decisons due to the danger of overfitting.

The study, [64], conducted by Hua et al. introduced models of a deep belief network and a deep convolutional neural network(CNN) as the first research to apply deep learning (DL) techniques to the problem of pulmonary nodule classification, which addressed the longstanding fundamental feature extraction problem for classification of the malignant or benign nature of lung nodules without actually computing the morphology and texture features. These two techniques achieved better discriminative results and proved to hold some promise in the computer-aided diagnosis (CAD) application, since they simplified the image analysis pipeline of CAD in the context of nodule classification in computed tomography (CT) images. In [45], a single CNN which was trained end-to-end directly on a dataset of 129,450 clinical images, consisting of 2,032 different diseases, using only the pixels and disease labels as inputs. This achieved a performance at par comparable to the of level of competence and expertise of dermatologists in their classification of skin lesions and cancer. The performance of the CNN was evaluated against 21 board-certified dermatologists on biopsy-proven clinical images with two critical binary classification use cases: keratinocyte carcinomas versus benign seborrheic keratoses; and malignant melanomas versus benign nevi. CNN's have further proven to be accurate with classification tasks such as in the classification of tuberculosis on chest radiographs with further improved accuracy after utilizing a radiologist-augmented approach [76]

However, recent studies [29], [55], [83] have shown that transfer learning has become a popular technique for training machine learning classifiers especially for medical imaging, where datasets can be relatively small. Tajbakhsh et al. [123] answer the question of using pre-trained deep CNN's with sufficient fine-tuning as an alternative technique to eliminate the need for training a deep CNN from scratch. They review full training and fine-tuning with Alexnet pretrained on ImageNet in a layerwise fashion on a small unrelated type of data. They considered four distinct medical imaging applications (polyp detection in colonoscopy, image quality assessment in colonoscopy, pulmonary embolism detection in CT, and intima-media boundary segmentation in ultrasonography) in the areas of radiology, cardiology, and gastroenterology imaging modalities. The observations from their experiments indicated that the use of pre-trained CNN's with adequate fine-tuning outperform a CNN trained from scratch which was consistent in all their experimentations.

Similar result was reported in [63] which compared training from scratch, off-the-shelf and fine-tuning strategies for different networks: Cifarnet (trained on CIFAR-10), Alexnet (trained on Imagenet) and GoogLeNet on two tasks: thoraco-abdominal lymph node detection and interstitial lung disease classification. The overall findings from the study indicate that full training and fine-tuning lead to the best results, with fine-tuning being most beneficial for GoogLeNet. The off-the-shelf strategy was only applied to Cifarnet which gave the worst results. But Cifarnet outperformed Alexnet. Alexnet achieved similar performance with all the three strategies on the interstitial lung disease classification. Similar comparison is by Menegola et al. [89] in their report which compared off-the-shelf, full training and fine-tuning strategies for a VGG network. They addressed the melanoma classification in skin lesion images using Imagenet and Kaggle diabetic retinopathy (DR) as their source of data. The off-the-shelf features outperformed full training when transferring from Imagenet, performed comparably with full training when transferring from Kaggle DR. But in all, the overall conclusion from their report proves that fine-tuning outperforms off-the-shelf features when transferring from both sources. And this was in contrast to their initial hypothesis, which tipped Kaggle DR to lead to best results due the visual similarity of the data.

Different from the above papers discussed is the work of Ribeiro et al. [106], which investigated pre-training and fine-tuning with nine different sources of datasets. Cheplygina [28] provides an exhaustive summary of several papers which use medical or non-medical data as source data and apply the classifier on medical target data. Transfer learning strategies, which involve fine tuning of a network pre-trained on different dataset, have recently been actively applied to a variety of classification and identification tasks such as in [130], [134] and [76] respectively. Most of these studies [31], [75], [139], [140] employed the transfer learning concept which combines fine-tuning and deep features approach to obtain more task-specific deep feature representations by replacing and retraining the deepest layer of a network, whereas shallow layers are fixed after the initial training [87].

3. The Intelligent Conception

This chapter briefly describes the fundamental basics of neural networks. We describe the feed-forward deep neural network model and later present the regularization and optimization techniques of the deep network model. We further describe the convolutional neural network as the specialization technique for scaling the deep network model to large image classification task. Finally, we present the concept of transfer learning from selected state-of-the-art model architectures from ImageNet Large Scale Visual Recognition Competition (ILSVRC) to be adopted in the design of our patient model for decision support based on medical radiology image classification.

3.1 Machine Learning

Machine learning is an extensive domain that can be applied in several different fields. In this work, it is meant to be understood as a body of intelligent concepts from which emphasis is made on deep learning to build medical image classification model to function as the reasoning inference component for our clinical decision support system.

Machine learning is a computer program that learns from experience with respect to some class of tasks and performance measure, if its performance at the given tasks as measured by the performance improves with experience [92]. Machine learning algorithms are said to learn when they are able to improve their performance after they have attained the ability to perform their given task through their experience on a given set of data to gain knowledge and expertise. Since the learning activity of machine learning algorithms involves an interaction between the algorithm and the data, the quality of input data determines the performance of the learner.

In the context of our work, learning tasks are classified according to the nature of their interactions. Machine learning can be supervised, semi-supervised or unsupervised and re-enforcement. In a healthcare facility, where sample of sputum or mucus is inspected, pulse oximetry estimated, blood test and pressure taken and finally chest X-ray inspected of newly admitted patients. It is required that decisions are made as to whether to put the newly admitted patients on Lower Respiratory Tract Infections (LRTI) treatment such as mechanical ventilation support (CPAP, BiPAP etc), in which case they are categorized as high-risk patients or use a bronchodilator inhaler for patients categorized as low-risk with higher priority given to those with respiratory insufficiency.

Now, the task in such an instance is to predict high-risk patients and sergregate them from low-risk patients based on their recorded information. A machine learning algorithm learns from the historic health records (data) of patients in the facility who are labeled with pre-defined classes, which represent some "past experiences". The learner learns a target or decision function that can be used to predict the values of a discrete class attribute which classifies the new patients as high-risk or low risk. This task is called supervised learning. In unsupervised learning, algorithms are presented with data that are not labeled. Instead, the algorithm is tasked with identifying patterns in the data on its own by defining signals and potential abnormalities based on the clustering of data. Semi-supervised learning combines the two scenario of supervised where some of the data are labeled and the scenario where part of the data is unlabeled.

Machine learning is heavily focused on using special abilities of computers to complement human intelligence, often by performing tasks that are beyond human capabilities such as scanning and processing huge medical image databases to identify patterns and classify medical images that are often outside the scope of human perception and capability due to time constraints. It offers techniques for predictive analytics to derive insights from data by analyzing historic and current data to predict the future.

3.1.1 General Structure of Machine Learning Algorithms. Generally, all machine learning algorithms are built based on the following four basic structure irrespective of their adoption of different unique techniques. These components are (1) Data representation (2) Model (3) Cost function (4) Optimization technique. The structure of all machine learning algorithms described in this work follow this laid out convention.

3.1.2 Data representation. For all machine leaning tasks, it is important to have a representation of the input data with specific properties that a computer can understand. Good data representation serves as the hypothesis space from which machine learning algorithms gain useful information, and without which they can not learn.

Using the above hospital supervised classification task as an illustration, a classifier would input a vector of discrete and or continuous feature values and output a single discrete value, the class, i.e. it classifies the patients into "high-risk" or "low-risk". The data may contain a set of n - instances or patient observations which comprises a set of normalized input features and the output or target class. The input features and target output labels are represented as an input vector X and output vector y respectively.

$$X_i^T = [x_1, x_2, x_3 \cdots x_n] \tag{3.1.1}$$

$$y_i^T = [y_1, y_2, y_3 \cdots y_n], \tag{3.1.2}$$

where i is the instance of a patient and n is the total number of patient observations. A learner inputs a sample of the set of pairs (X_i^T, y_i^T) as the training set and the remaining set of the data is used as the test set.

3.1.3 Model. Model is a mathematical prediction formulation or equation that defines the relationship between how input dataset X_i predict the value of it's corresponding output response y_i during the training process. Models are built from parameters represented by vectors θ, which are adjusted during the training process. Machine learning algorithms rely on the use of decision functions to model real life problems in the form of mathematical equations, trees or rules for knowledge discovery. This gives rise to the popular saying by Dr. Box, "All models are wrong. But some are useful.", since it is impossible to model real life problems perfectly with mathematical equations.

The simplest model, linear model also called linear regression makes a prediction $\hat{y}_i$ on an instance i of n number of input features,X, using the model formulation;

$$\hat{y} = X\theta + \epsilon \tag{3.1.3}$$

$$\hat{y} = \textstyle\sum_i^n X_i\theta_i + \epsilon \tag{3.1.4}$$

With matrix notation:

$$\begin{bmatrix} \hat{y}_1 \\ \vdots \\ \hat{y}_n \end{bmatrix} = \begin{bmatrix} x_{11} \cdots x_{1n} \\ \vdots \\ x_{n1} \cdots x_{nn} \end{bmatrix} \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} \tag{3.1.5}$$

3.1.4 Loss or Cost function. The cost function measures the deviation of a model in terms of its ability to estimate the relationship that exist between the input features and the target variable. It serves as a judge in our preference of one model to another. An evaluation function also called objective function or scoring function is needed to distinguish good classifiers from bad ones. The evaluation function used internally by the algorithm may differ from the external one that we want the classifier to optimize, for ease of optimization.

3.1.5 Optimization technique. Finally, we need a method to search among the classifiers in the language for the highest-scoring one. The choice of optimization technique is key to the efficiency of the learner, and also helps determine the classifier produced if the evaluation function has more than one optimum.

3.1.6 Deep Learning. Deep Learning is a subset of machine learning which provides very powerful framework to remedy critical limitations of other types of machine learning approaches since its origination from artificial neural networks in the 1950. It functions as a type of representation learning in which no feature engineering is used; instead, computers are made to learn the features by which to classify the provided data inputs. It is extensively used in applications such as computer vision, that have to do with huge image classification, recognition, detection etc. It is among the most successful research areas in the recent times since the advent of big data. In this context, deep learning is a term that encapsulates the study of neural nets with many hidden layers related to the family of supervised and unsupervised learning methods, based on artificial neural networks.

3.2 Artificial Neural Networks (ANN)

The human brain is made of billions of neurons which communicate with one another through the use of electrical impulses to create synapses [51]. A neuron is a biological cell and forms the foundational unit block of the human central nervous system (CNS). It is composed of biological networks interconnected with other neurons and are optimized to receive and send information (signals) to other connected neurons. The neuron receives its inputs along antennae-like structure of fibres called dendrites. Information in the form of signals are propagated along the cell's axon and sent off to other neurons after weighted by the strength or weakness of the connection based on the frequency of usage and summed in the cell body (see figure 3.1). The function of the nervous system depends on groups of neurons that work together. Individual neuron connects to other neurons to form networks that may be very simple and made of few neurons or may involve very complex neuronal networks that process incoming (signals) information and carry out a response.

The above biological process has been the background inspiration behind the build up of mathematical models for artificial neurons. This was first proposed by Warren Mcculloch and Walter Pitts in 1943 [88]. Artificial Neural Networks (ANN) are built by using basic unit blocks called perceptons [91]. They form a class of models which are arranged in a way analogous to how real neurons are connected. Just as real neurons fire after reaching a certain activation threshold, the individual nodes of a neural net each represent an activation function triggered by a certain level of input. They imitate the functional process of certain parts of the biological neurons, such as the dendrites, cell bodies and axons using simplified mathematical models. The models employ the use of a function that receives a list of weighted input signals and outputs signals. The weights are the basic means of long-term memory in ANNs. They express the strength of importance of each neuron input.

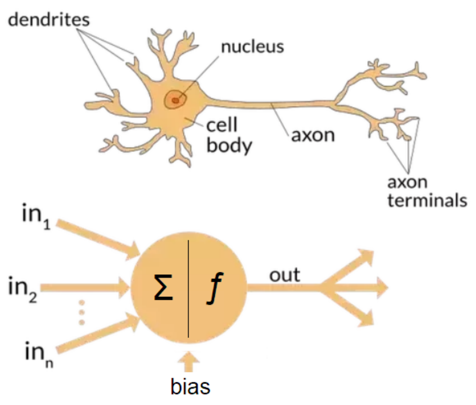


Figure 3.1: biological vrs artificial neuron (https://www.quora.com)

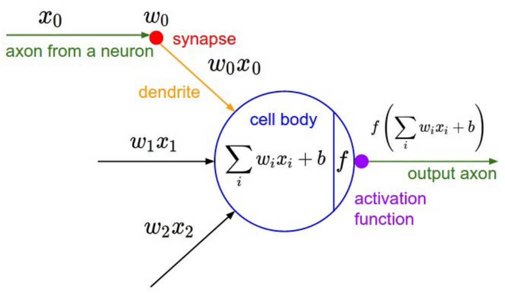


Figure 3.2: Typical schema of artificial neuron composed of inputs, weights, bias and activation function

3.3 Structure of Artificial Neural Networks

An artificial neural network (ANN) is arranged in layers. The first layer is called the Input layer and contains the input vector of the data representation. Each node in the input layer represents an input variable in the model. And then followed by one or more hidden layers, which can each have any specified number of hidden nodes depending on how deep the neural network is built. The output layer is the final layer with number of nodes depending on the desired output response. The layers are hierarchically stacked and each neuron inside the hidden layers has input connections from all neurons of a previous layer. The output of a previously connected neuron becomes the input of each neuron in a subsequent hidden layer represented by the directed connections. The hidden layers are distinguished from each other only by the choice of the form of the activation function. This characterizes the forward flow of information from inputs to output during activation. Hence, the name Feed-forward Fully Connected Neural Network. Figure 3.3 shows a typical Feed-Forward Fully Connected Artificial Neural Network with two hidden layers.

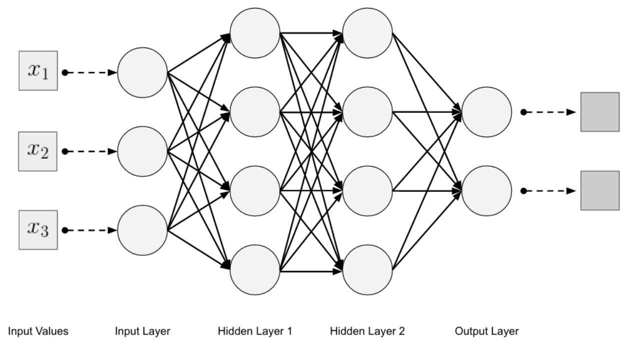


Figure 3.3: Artificial Neural Net

3.3.1 Activation Functions. An activation function makes a non-linear transformation of its hypothesis space (inputs), and grants ability to the network to estimate or approximate complex functions. The activation function takes the sum of the weighted inputs as an argument and return an output to the neuron. Mathematically, this is expressed as :

$$a^i = g\big(\sum_i (w_k^i.x_k^{i-1}) + \beta^i\big) \ , \tag{3.3.1}$$

$$z^i = \sum_i (w_k^i.x_k^{i-1}) + \beta^i \tag{3.3.2}$$

$$a^i = g(z^i) \tag{3.3.3}$$

where:

w_i^k is the weight from the k^{th} neuron in the $(i-1)^{th}$ layer to the k^{th} neuron in the i^{th} layer

β^i is the bias of the k^{th} neuron in the i^{th} layer

a^i represents the activation value of the k^{th} neuron in the i^{th} layer.

There exist different types of activation functions which are commonly used. Among these are sigmoid function (popularly known as the logistic function), hyperbolic tangent $(tanh)$, softmax and rectified linear unit (RELU).

3.3.2 sigmoid function. The sigmoid function belongs to the larger class of algorithms known as generalized linear model (GLM). It operates on the assumption of a linear relationship between a link function and the set of independent features. It is used in classification tasks to predict a binary or categorical outcome ie. 1 / 0, Yes / No, True / False. The sigmoid function predicts the probability that a neuron is activated by fitting data to a logit function. The major setback of this activation function is the problem of vanishing gradient which arises as the activation approaches the horizontals on either sides of the sigmoid curve. However, it has the advantage of not blowing up the activation since its output lies between 0 and 1. The formulation of sigmoid's logit function emanates from the fundamental generalized linear model (GLM) which makes use of a linear function in a link function mathematical expressed as:

$$f(x) \ = \ wx + \beta = z \tag{3.3.4}$$

The above link function assumes that:

- probability of success, p, is always positive and lies between 0 and 1.
- probability of failure is given as $1-p$.

Since probability must always be positive, this gives rise to the problem of vanishing gradients.

We denote $g(z)$ with p and then put the linear equation (3.3.4) in exponential form

$$p \ = \ \exp{(wx + \beta)} \tag{3.3.5}$$

and then divide p by a number greater than p in order to satisfy the first assumption of the link function.

$$p \ = \ \frac{\exp{(wx+\beta)}}{\exp{(wx+\beta)}+1} \tag{3.3.6}$$

Hence, the probability of failure, $q = 1 - p$ is given as:

$$q \;=\; 1 - \frac{\exp{(wx + \beta)}}{\exp{(wx + \beta)} + 1} \tag{3.3.7}$$

We divide p by q to obtain the odd ratio:

$$\frac{p}{q} = \frac{p}{1-p} = \frac{\dfrac{\exp{(\alpha + \beta x)}}{\exp{(wx + \beta)} + 1}}{1 - \dfrac{\exp{(wx + \beta)}}{\exp{(wx + \beta)} + 1}} = \exp{(z)} \tag{3.3.8}$$

We take logarithm of both sides and this yields:

$$\log{(\frac{p}{1-p})} = \log{(\exp{(z)})} \tag{3.3.9}$$

$$\log{[\frac{p}{1-p}]} = z \tag{3.3.10}$$

The sigmoid function obtained is S-Shaped (see figure 3.4) and defined as:

$$\phi(z) = \frac{1}{1 + \exp{(-z)}} \tag{3.3.11}$$

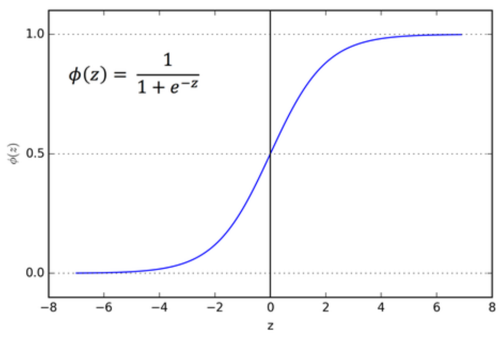


Figure 3.4: sigmoid function

3.3.3 Hyperbolic tangent (tanh). The hyperbolic tangent $(tanh)$ is a scaled version of the sigmoid function but symmetric about the origin. It is defined as

$$a = g(z) = tanh(-z) \tag{3.3.12}$$

$$a = g(z) = \frac{\exp{(z)} - \exp{(-z)}}{\exp^{(z)} + \exp{(-z)}} \tag{3.3.13}$$

$$a = \frac{2}{1 + \exp(-2z)} - 1 \tag{3.3.14}$$

$$a = 2\phi(2z) - 1, a \in [-1, 1] \tag{3.3.15}$$

It is predominantly used in recurrent neural networks with less usage in feed-forward neural networks. It is nonlinear in nature but also has the drawback of vanishing gradient even though its gradient is steeper as compared to the sigmoid.

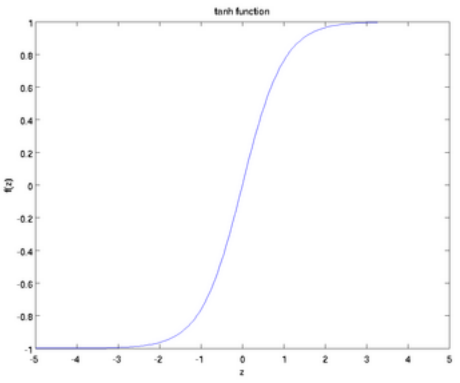


Figure 3.5: Hyperbolic tangent (tanh)

3.3.4 Softmax. The softmax function is a generalization of the sigmoid function. It is basically the normalized exponential probability of class observations represented as neuron activations. It is conventionally used as the classification layer in most multi-class deep learning models. It fulfils the constraints of a probability density by ensuring that the logits all sum up to 1. It specifies a discrete probability distribution for say m classes denoted by:

$$\textstyle\sum_{i=1}^{m} p_i \tag{3.3.16}$$

If x is the activation at a neuron within the hidden layer of a neural network, and w is its weight parameter at the softmax layer,then we have z as the input to the softmax layer,

$$z = \textstyle\sum_{i=1}^{n-1} w_i x_i \tag{3.3.17}$$

Hence,

$$p_i = \frac{exp(z_i)}{\sum_{i=1}^{n-1} exp(z_i)} \tag{3.3.18}$$

This essentially gives the probability of the input being in a particular class. Hence, yielding the predicted class $\hat{m}$ as:

$$\hat{m} = argmax(p_i), \quad i \in 1, \cdots m \tag{3.3.19}$$

3.3.5 Rectified Linear Unit (ReLU). ReLU is a simple function defined as

$$g(z) = max\{0, z\} \tag{3.3.20}$$

It is the most used activation function especially in convolutional and fully connected layers. This is due to its unique ability to convert negative inputs into zero which translates into the automatic deactivation of the neuron involved. The ReLU function (Figure 3.6) does not activate all the neurons at the same time. Only few neurons get activated at specific time periods ensuring computational efficiency, and this helps it to learn quickly. However, it has a drawback which makes it not differentiable over the entire domain for z = 0. An improved form called the Leaky ReLU [84] helps to train deeper networks and its defined as:

$$LReLU(z) = \begin{cases} z & if \quad z \geq 0 \\ \alpha z & otherwise, \end{cases} \tag{3.3.21}$$

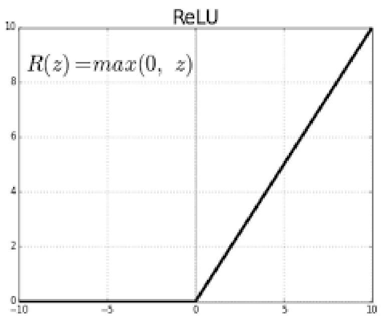


Figure 3.6: ReLU

We briefly describe the feed-forward flow of information from inputs to output during activation using an ANN with a single hidden layer. Figure 3.7 summarizes this description of the feed-forward process.

The ANN performs feed-forward computations by multiplying the input feature vectors, a_i, from the input layer by a set of fully-connected weights w_{ij} that connect the input layer to the hidden layer. The weighted features are summed and combined with a bias term β_i located on each neuron in the hidden layer to form pre-activation signal for the hidden layer:

$$z_j = \textstyle\sum_i a_i w_{ij} + \beta_j \ . \tag{3.3.22}$$

The pre-activated signal z_j is transformed into non-linear forms by the hidden layer's activation function g_j to form the feed-forward activation signals leaving hidden layer a_j. These signals from a_j are also multiplied by the weights connecting the next hidden layer to the output layer w_{jk}, a bias β_k is added as it was done previously. In the case of an ANN with many hidden layers, this process continues similarly for all the hidden layers within the network until the resulting signal from the last hidden layer is transformed by the output activation function g_k to form the network output y_k. The network's output y_k is then evaluated against the actual target t_k and the error between the two is computed as the cost or loss function.

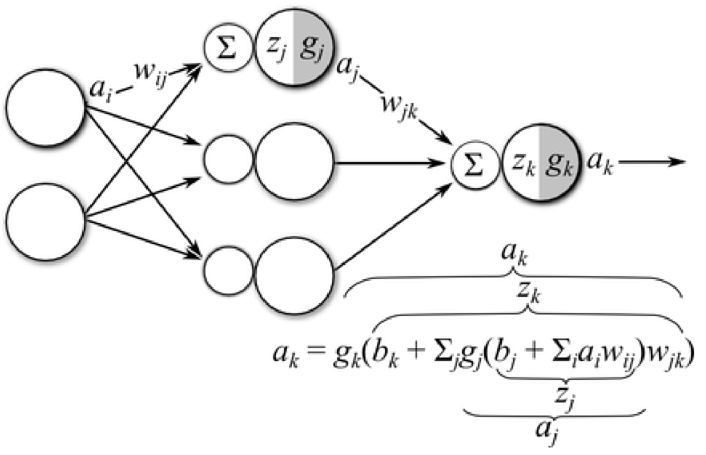


Figure 3.7: Illustration of feed-forward source: https://theclevermachine.wordpress.com

3.4 Cost Function

The Cost function of an artificial neural network measures how well the neural network performs in relation to the training sample at its disposal during the training process against the desired or actual output. The set of parameters $\theta = \{\mathbf{w}, \beta\}$ minimize the errors that the network makes. Typically, the cost function is written as an average (expectation) over the entire training data set. This is denoted by:

$$\hat{\mathcal{C}}(\theta) \;=\; \mathbb{E}_{(x,y)\sim\hat{P}data} C(f(x;\theta), y) \;, \tag{3.4.1}$$

where C is the per sample loss function,$f(x;\theta)$ is the predicted output when the input is x, $\hat{P}data$ is the empirical distribution.

The error function, E, for linear layers is often the sum of the squared difference between the target values t_k and the network output y_k, also called mean squared error (MSE). Mathematically, E is represented as:

$$E \;=\; \tfrac{1}{2}\textstyle\sum_{k\in K}(y_k - t_k)^2 \tag{3.4.2}$$

Cross-entropy with variants, binary and categorical, are the most used cost functions in neural networks for classifcation tasks in which sigmoid and softmax layers are used respectively. The binary cross entropy classifies into two distinct classes. The categorical is used for multiple classes. The categorical crossentropy (cost function) for the softmax layer is defined using equation (3.3.19) as:

$$\hat{\mathcal{C}} \;=\; -\frac{1}{N}\textstyle\sum_{i=1}^{N} y^{(i)}\ln p_i + (1 - y^{(i)})\ln(1 - p_i) \tag{3.4.3}$$

They are solved during the optimization of the associated error by using gradient descent algorithm. A few other cost functions include:

The Mean Absolute Error (MAE) measures the average magnitude of the errors in a set of predictions, without considering their directions. It's the average over the test sample of the absolute differences between prediction and actual observation. MAE gives a relatively high weight to large errors as it squares the error.

Mean Absolute Percentage Error(MAPE) measures the size of the error in percentage terms. It is calculated as the average of the unsigned percentage error. It is used because of the ease of interpretability of percentages.

Hinge loss/squared-hinge loss is used in Support Vector Machines (SVMs). It penalizes marginally misclassified points differently. It is a good alternative to cross-entropy loss and lead to faster training for neural networks as well. Higher-order hinge losses, such as squared-hinge losses, are even better for some classification tasks.

Kullback-Leibler (KL) divergence is a measure of how one probability distribution diverges from a second expected probability distribution.

3.5 Optimizing The Cost Function

Optimization techniques employed in deep learning are based on the gradient descent algorithm [23], [79]. Gradient descent helps to achieve the aim of minimizing the deviation of the output of the neural network from the actual output. It does so by minimizing the weights of the neurons that contribute much to the deviation of the network output. This is achieved while traveling back to the neurons of the network where the error resides through the process called backward propagation popularly known as backpropagation [62], [114].

3.5.1 Backward Propagation. The backpropagation algorithm is used to find a local minimum of the error function E. It is used to compute the necessary corrections. The network is initialized with randomly chosen weights. The gradient of E is computed making use of the chain rule ie. partial derivatives $\frac{E}{\delta w_i}$ of the error function, E, with respect to the weights, w_i, in the network and are used to correct the initial weights. The parameters, θ, in the network are modified to make the error, E, as low as possible. This functions against the assumptions that:

- cost function is a function of the neural network's output.
- total cost function averages over all cost functions for all individual training samples

The backward propagation algorithm looks for the minimum of the error function E in weight space using the method of gradient descent.

$$\nabla_E = (\frac{\partial_E}{\partial_{w1}}, \frac{\partial_E}{\partial_{w2}}, \cdots, \frac{\partial_E}{\partial_{wN}}) \tag{3.5.1}$$

Since this method requires computation of the gradient of the error function E at each iteration step, activation functions that are continuous and differentiable are used. The algorithm begins at the cost function of the output layer of the neural network.

$$E^l = \nabla_w \mathcal{C} \odot g^{'}(z^L) \tag{3.5.2}$$

where

l is the last layer of the network.

$\nabla_w \mathcal{C}$ is gradient of cost function with respect. to w.

$\odot$ is the Hadamard product and

g is the activation function.

The errors from subsequent lower layers are computed as:

$$E^l = ((w^{l+1})^T E^{l+1} \odot g^{'}(z^l) \tag{3.5.3}$$

where

$(w^{l+1})^T$ is obtained from a^i_j in eqn 3.3.1.

Each neuron modifies its weight w and bias β parameters estimated from a^i_j in eqn 3.3.1 respectively as:

$$\frac{\partial C}{\partial w^l_{jk}} = x^{l-1} E^l_j \tag{3.5.4}$$

Each weight is updated using the increment;

$$\Delta w_i = -\phi \frac{\partial E}{\partial w_i} \quad for \quad i = 1, \cdots, n \tag{3.5.5}$$

where ϕ represents a learning constant (rate) which determines how quickly the parameters are updated, i.e., a proportionality parameter which defines the step length of each iteration in the negative gradient direction.

It is to be noted that back-propagation refers only to the method for computing the gradients. Other algorithms, such as stochastic gradient descent and its variants, are used to perform learning using these gradients.

3.5.2 Stochastic Gradient Descent(SGD). SGD provides an unbiased estimate of the gradient by taking the average gradient on a minibatch of M samples drawn from data which is generated identically and independently from the data generating distribution. The learning rate ϕ is the most important parameter in SGD and it decreases gradually over time as the true gradient of the total cost function approaches zero 0 ie. minimum.

Considering a simplier linear unit, the SGD algorithm could be summarized as:

$$y = w_0 + w_1 x_1 + \cdots + w_M x_M \tag{3.5.6}$$

Let's learn w_i's that minimize the squared error.

$$E[\vec{w}] \equiv \tfrac{1}{2} \textstyle\sum_{d \in M} (t_d - y_d)^2 \tag{3.5.7}$$

Gradient

$$\phi E[\vec{w}] \equiv \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \cdots \frac{\partial E}{\partial w_M} \right] \tag{3.5.8}$$

Training rule:

$$\Delta\vec{w} = -\phi E[\vec{w}] \tag{3.5.9}$$

i.e.,

$$\Delta w_i = -\phi \tfrac{\partial E}{\partial w_i} \tag{3.5.10}$$

Gradient Descent

$$\begin{aligned}
\frac{\partial E}{\partial w_i} &= \frac{\partial}{\partial w_i}\frac{1}{2}\sum_d (t_d - y_d)^2 \\
&= \frac{1}{2}\sum_d \frac{\partial}{\partial w_i}(t_d - y_d)^2 \\
&= \frac{1}{2}\sum_d 2(t_d - y_d)\frac{\partial}{\partial w_i}(t_d - y_d) \\
&= \sum_d (t_d - y_d)\frac{\partial}{\partial w_i}(t_d - \vec{w}\cdot\vec{x_d}) \\
\frac{\partial E}{\partial w_i} &= \sum_d (t_d - y_d)(-x_{i,d})
\end{aligned}$$

Each training example is a pair of the form $\langle\vec{x}, t\rangle$, where $\vec{x}$ is the vector of input values, and t is the target output value. y is the predicted output value and ϕ is the learning rate (e.g., .05).

A sufficient condition to guarantee the concergence of SGD is that;

$$\textstyle\sum_{k=1}^{\infty}\phi_k = \infty \quad and \quad \sum_{k=1}^{\infty}\phi_k^2 < \infty \tag{3.5.11}$$

Many other variations of the gradient descent algorithm exist some of which have adaptive learning rates. A few of them are taken from [20], [54] and are briefly described as below.

3.5.3 Momentum. The method of momentum was introduced to accelerate the learning of SGD. It uses a variable v that plays the role of velocity with a hyperparameter $\alpha \in [0, 1)$. The momentum determines the rate at which the contributions of previous gradients exponentially decay. The update rule is given by:

$$\begin{aligned}
v &\leftarrow \alpha v - \phi\nabla_\theta\Big(\frac{1}{M}\textstyle\sum_{i=1}^{M}\mathcal{C}(f(x^{(i)};\theta), y^{(i)})\Big), \\
\theta &\leftarrow \theta + v
\end{aligned} \tag{3.5.12}$$

3.5.4 Nesterov Momentum. Sutskever et al. [119] introduced this as a variant of the momentum algorithm and was inspired by Nesterov's accelerated gradient method [95].

$$\begin{aligned}
v &\leftarrow \alpha v - \phi\nabla_\theta\Big(\frac{1}{M}\textstyle\sum_{i=1}^{M}\mathcal{C}(f(x^{(i)};\theta + \alpha v), y^{(i)})\Big), \\
\theta &\leftarrow \theta + v
\end{aligned} \tag{3.5.13}$$

3.5.5 AdaGrad. Duchi et al. [43] describe AdaGrad as a technique that adapts the learning rates of all model parameters by scaling them inversely proportional to the square root of the sum of all of their historical squared values. AdaGrad shrinks the learning rate according to the entire history of the squared gradient.

$$g \leftarrow \phi \nabla_\theta \big(\frac{1}{M} \textstyle\sum_{i=1}^{M} \mathcal{C}(f(x^{(i)};\theta), y^{(i)})\big) \tag{3.5.14}$$

$$r \leftarrow r + g \odot g \tag{3.5.15}$$

$$\Delta\theta \leftarrow -\frac{\phi}{\delta + \sqrt{r}} \odot g \tag{3.5.16}$$

$$\theta \leftarrow \theta + \Delta\theta \tag{3.5.17}$$

where

δ is a small constant about 10^{-7}

r initialize gradient accumulation variable, $r = 0$

3.5.6 RMSProp. RMSProp [61] is an adaptive learning rate method that divides the learning rate ϕ by an exponentially decaying average of the squared gradients. This Keeps a moving average of the squared gradient for each weight.

$$g \leftarrow \phi \nabla_\theta \big(\frac{1}{M} \textstyle\sum_{i=1}^{M} \mathcal{C}(f(x^{(i)};\theta), y^{(i)})\big) \tag{3.5.18}$$

$$r \leftarrow \rho r + (1-\rho) g \odot g, \quad \rho \quad is\ decay\ rate \tag{3.5.19}$$

$$\Delta\theta \leftarrow -\frac{\phi}{\delta + \sqrt{r}} \odot g \tag{3.5.20}$$

$$\theta \leftarrow \theta + \Delta\theta \tag{3.5.21}$$

3.5.7 Adam. Kingma and Ba [71] describe Adam as a variant of the adaptive learning rate algorithms whose named was derived from the phrase "Adaptive moments". It is obtained from the combination of RMSProp and momentum.

$$\hat{\theta} \leftarrow \theta + \alpha v \tag{3.5.22}$$

$$g \leftarrow \phi \nabla_{\hat{\theta}} \big(\frac{1}{M} \textstyle\sum_{i=1}^{M} \mathcal{C}(f(x^{(i)};\hat{\theta}), y^{(i)})\big) \tag{3.5.23}$$

$$r \leftarrow \rho r + (1-\rho) g \odot g \tag{3.5.24}$$

$$\Delta\theta \leftarrow -\frac{\phi}{\delta + \sqrt{r}} \odot g \tag{3.5.25}$$

$$v \leftarrow \alpha v - \frac{\phi}{\sqrt{r}} \odot g \tag{3.5.26}$$

$$\theta \leftarrow \theta + v \tag{3.5.27}$$

3.6 Regularization Techniques

The ultimate goal of all Machine Learning models is to generalize beyond the training data. We sometimes hallucinate our model when we do not have sufficient data, hence, encoding random twists leading to the problem of overfitting. In this case, the model learns too well the training data but performs badly on unseen data. Another issue called underfitting may as well result when it performs badly on the training data. Many methods such as cross-validation is mostly applied in Machine Learning to mitigate these problems. In addition to the above is the curse of dimensionality. This leads to algorithms becoming intractable when the number of input features grow. The collective name for the strategies employed to reduce the test error at the expense of increased training error is called regularization.

Some of the regularization techniques mostly employed by ANN in deep learning include but not limited to:

- Parameter Norm Penalty
- Data Augmentation
- Dropout
- Early Stopping

3.6.1 Parameter Norm Penalty. Denoted $\Omega(\theta)$ is a regularization term which is added to the cost (objective) function to limit the complexity of models. In the case of a neural network, the general cost function, $\mathcal{C}$, is updated and becomes:

$$Costfunction = Loss(eg.binary\_crossentropy) + regularization\ term$$

Mathematically this can be written as:

$$\hat{\mathcal{C}}(\theta; X, y) = \mathcal{C}(\theta; X, y) + \alpha\Omega(\theta) \tag{3.6.1}$$

where;

$\alpha \in [0, \infty)$ is the weight of the relative contribution of the norm penalty term $\Omega(\theta)$.

The choice of parameter norm Ω in neural networks is to only penalize the weights at each layer while leaving the biases unregularized. The L^2 parameter norm penalty (weight decay) adds a regularization term defined by:

$$\Omega(\theta) = \frac{1}{2}\|w\|_2^2 \tag{3.6.2}$$

to the cost (objective) function. Hence, the regularized cost function now becomes

$$\hat{\mathcal{C}}(\theta; X, y) = \mathcal{C}(\theta; X, y) + \frac{1}{2}\|w\|_2^2 \tag{3.6.3}$$

Assuming no bias parameter, hence, θ is just w. The total cost (objective) function now becomes:

$$\hat{\mathcal{C}}(w; X, y) = \mathcal{C}(w; X, y) + \frac{1}{2}\|w\|_2^2 \tag{3.6.4}$$

equivalently written as:

$$\hat{\mathcal{C}}(w; X, y) = \mathcal{C}(w; X, y) + \frac{\alpha}{2}w^T w \tag{3.6.5}$$

The corresponding gradient is:

$$\nabla_w \hat{\mathcal{C}}(w; X, y) = \nabla_w \mathcal{C}(w; X, y) + \alpha w \tag{3.6.6}$$

During backpropagation, the weights are updated in a single step as;

$$w \leftarrow w - \epsilon\big(\alpha w + \nabla_w \mathcal{C}(w; X, y)\big) \tag{3.6.7}$$

$$w \leftarrow (1 - \epsilon\alpha)w - \epsilon\nabla_w \mathcal{C}(w; X, y)\big) \tag{3.6.8}$$

The weight decay shrinks the weight vector by a constant factor through each step, and this helps to improve the learning process of the model. L^2 parameter norm penalty regularization is popularly called the ridge regression or Tikhonov regularization.

In the case of an L^1 parameter norm penalty, the regularization term is defined as:

$$\Omega(\theta) = \|w\|_1^1 = \textstyle\sum_i |w|_1^1 \tag{3.6.9}$$

which results in a regularized cost function:

$$\hat{\mathcal{C}}(\theta; X, y) = \mathcal{C}(\theta; X, y) + \textstyle\sum_i |w|_1^1 \tag{3.6.10}$$

The parameter norm penalty regulates the contribution of the regularization by scaling the penalty $\Omega(\theta)$ using a positive hyperparameter α. High values of α indicate more regularization and smaller values indicate the versa. Generally, the regularization terms cause the values of the weight matrices to shrink since it assumes that the neural networks with smaller matrices have smaller models.

3.6.2 Data Augmentation. The strict privacy concern of data eg. medical data, imposes huge data protection mechanisms on their accessibility and acquisition. This has created artificial limitation to access on the public and further resulted in insufficient data available during medical image classification experiments. The high affinity of deep learning models for data to improve their generalization performance gave way to the birth of data augmentation as a means of creating 'fake' data to augment this insufficiency.

Van Dyk et al. [127] define data augmentation as the methods for constructing iterative optimization or sampling algorithms via the introduction of unobserved data or latent variables. This is an effective technique for a specific classification tasks such as object (image) recognition. Operations such as cropping, rotation, scaling and flipping are applied to an input data (image) to translate a few pixels of it in directions (right, left, up, down etc) specified by angles in the algorithm to generate new fake ones of the original input. The invariant property of most natural images to translation motivate this technique.

3.6.3 Dropout. The fundamental concept of dropout in neural networks is to randomly drop units of neurons in hidden layers temporarily along with their incoming and outgoing connections during the training process. Dropout uses a tunable hyperparameter p (the probability of retaining a unit in the network) to sample "thinned" network from the entire neural network which is trained using backpropagation. This avoids the mechanism of training all neurons of layers together as a family by

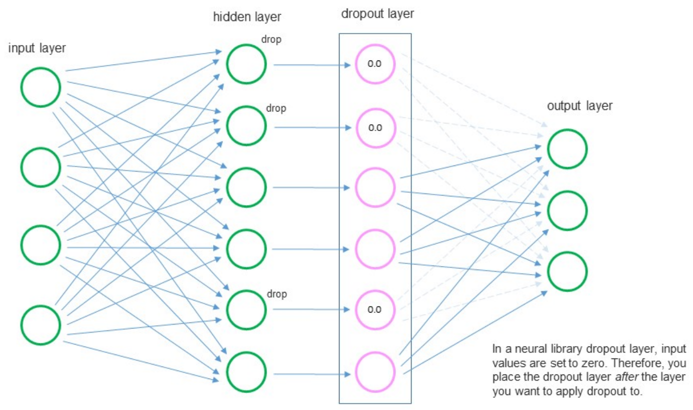

hidden layer
dropout layer
input layer
drop
0.0
drop
0.0
output layer
drop
0.0
In a neural library dropout layer, input values are set to zero. Therefore, you place the dropout layer *after* the layer you want to apply dropout to.

Figure 3.8: Dropout

deactivating some of the neurons which reduces co-adaptability. Training a network with dropout is likened to an ensemble approach of training collections of thinned neural networks. Individual activated neurons within a thinned network learn to take responsibility which improves its performance.

At test time, a single neural net with smaller weights is used without dropout to average the responses of all thinned networks at training. This significantly reduces overfitting and gives major improvements over other regularization methods.

3.6.4 Early Stopping. Early stopping is simply stopping the training process as soon as the performance on a validation set starts to get worse. Early stopping has the effect of restricting the optimization procedure to a relatively small volume of parameter space in the neighborhood of the initial parameter value [22]. Every time the error on the validation set improves, we store a copy of the model parameters.When the training algorithm terminates, we return these parameters, rather than the latest parameters [54] It's controlled by a patience hyperparameter, which sets the number of times to observe

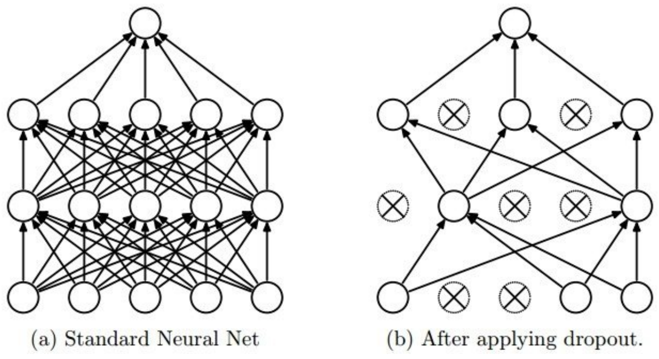


Figure 3.9: After applying dropout

increasing validation set error before training is aborted. It can be used alone or with other regularization techniques.

3.7 Convolutional Neural Networks

Convolutional neural networks [78] are the greatest success story of biologically inspired artificial intelligence [54]. They are specialized neural networks that imitate the neurons in the visual cortex of mammals with their core design principles drawn from neuroscience. They are mostly referred to as ConvNets or CNNs and have architectures that process data based on the explicit assumptions that the input data have grid-like topology (vector-valued observations). eg. images, time series data etc. This makes CNNs stand out in computer vision due to their contribution to the state-of-the-art breakthroughs that are influencing deep learning as a result of the neuroscientific principles. CNN employs specialized linear operations called convolution in at least one of its layers. This operation enables the CNN to learn higher-order features in the data by ensuring sparse interactions, parameter sharing and equivariant representations to improve the machine learning system. Goodfellow and Bengio [54] elaborate more on these and provide an extensive coverage of the convolution operation.

In a typical multilayer convolutional network, each layer of the network consists of three stages. In the first stage, the layer performs several convolutions in parallel to produce a set of linear activations. In the second stage, each linear activation is run through a nonlinear activation function, such as the rectified linear activation function. This stage is sometimes called the detector stage. In the third stage, we use a pooling function to modify the output of the layer further [54]. The input to the second layer is the output of the first layer, which usually has the output of many different convolutions at each position.

3.7.1 Architecture of CNN. Convolutional Neural Networks take advantage of the fact that the input data consists of images. Hence, they constrain the architecture in a manner typically composed of three different types of layers. These layers are: Convolutional Layer, Pooling Layer and Fully-Connected Layer. The convolutional and pooling layers make up the feature extraction part of the CNN architecture and the fully connected layer is the classification part of the architecture. These layers are stack to form the full CNN architecture. Each type of layer has its own rules for forward and backward error signal propagation.

3.7.2 Convolutional Layer (CONV). The convolutional layer functions as the core building block of a CNN and performs most of the computational heavy lifting through the use of convolution as a linear operation. The convolution operation is performed on input data (pixels of the image) with learnable

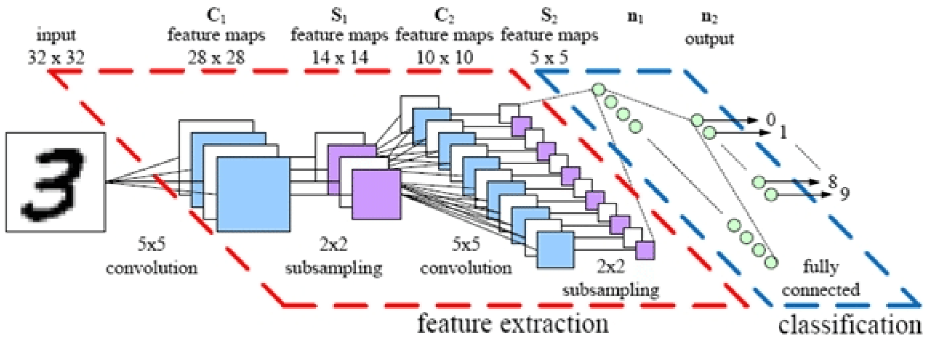


Figure 3.10: Typical Architecture of CNN

filters also called kernels. Kernels and filters are used interchargeably for the remaining part of this work. Kernels are typically square matrices with their width (mostly odd) ranging from 3 to N pixels that extend through the full depth (the color channels) of the input volume. The convolutional layer primarily functions as the layer responsible for the extraction of relevant input features by using the several filters that it is composed of. The output from the convolution operations generate feature maps. For example, a typical filter on a first layer of a CNN might have size $(5 \times 5 \times 3)$ (i.e. 5 pixels width and 5 pixels height, and 3 because images have depth 3, the color channels).

During the forward pass, we slide each filter across the width and height of the input volume (ie. pixels of the image) and compute dot products between the entries of the filter and the input at any position (see figure 3.10). As we slide the filter over the width and height of the input volume we will produce a 2-dimensional activation map that gives the responses of that filter at every spatial position [8]. Due to the high-dimensional nature of input image data, it is impractical to connect neurons to all the neurons in the previous volume. Hence, each neuron is connected to only a local region of the input volume through the use of a defined filter size which is entered as a hyperparameter called the receptive field of the neuron. The number of input pixels to be traversed when moving to neighboring position in each step by a filter is determined by the stride. A stride of 1 moves the filters one pixel at a time. When the stride is 2 or more, then the filters traverse 2 or more pixels respectively at a time as we slide them across (see figure 3.10). This produces smaller output volumes spatially. When the spatial size of the output feature map is desired to be preserved, it is always convenient to pad the input volume around the borders or edges. Zero padding adds necessary amount of zero elements around the borders (edges) of the input volume, it is entered as a hyperparameter with different variants such as full, valid and same.

The spatial size of the output volume from the convolutional layer is mathematically expressed as a function of the input volume (I), filter size (F), number of strides to be applied (S) and the number of padding (P) to be applied on the edges:

$$\frac{I - F + 2P}{S} + 1 \tag{3.7.1}$$

where

$$P = \frac{F-1}{2} \tag{3.7.2}$$

It is very important to note that convolutional layer derived its name from the computation of the neuron's weights with the input volume since all neurons in a single depth slice make use of the same

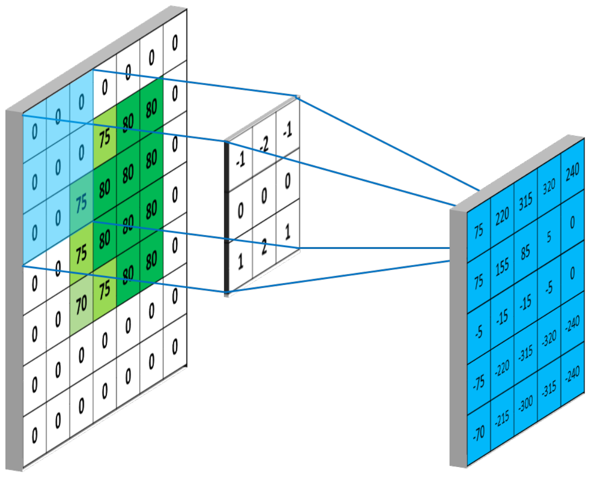


Figure 3.11: Illustration of input image pixels with zero padding that is being convolved into feature map using a filter

weight vector. This is why we refer to the sets of weights as a filter (kernel), that is convolved with the input.

3.7.3 Pooling Layer. Pooling is done for the purposes of reducing the spatial size of the input image data. It helps to make the representation become approximately invariant to small translations. The pooling layer compresses information contained within an input image data and preserves the most important features. Its function is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network, and perhaps, control overfitting. The most commonly used type of pooling is the max pooling [137]. The max pooling unit learns to become invariant by selecting the maximal value from learned linear combination of units within element. The L^2- norm pooling and the average pooling are other types of pooling approaches. However, it has

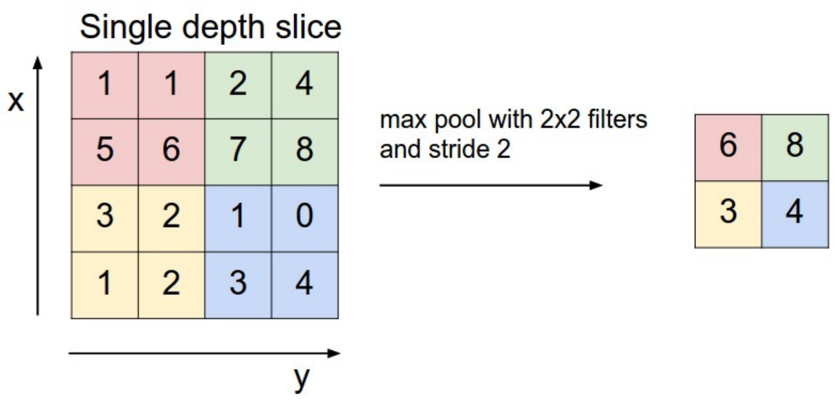


Figure 3.12: figure showing downsampling operation using max

been discovered in the studies [117] that the pooling layer (max-pooling) could simply be avoided or replaced by a convolutional layer with increased stride without affecting the accuracy on several image recognition benchmarks. Training generative adversarial networks (GANs) [136] and variational autoencoders (VAEs) have also proven to be effective without the use of pooling layers. hence, this poses the probable occurence of future CNN architectures built without pooling layers or with very few.

3.7.4 Fully Connected Layer. Fully Connected layer is identical to the layer from Feed-forward Fully Connected Neural Network (FCNN) discussed in section 3.3. Neurons in a fully connected layer have full connections to all activations in the previous layer, as seen in regular Neural Networks. Their activations are computed with a matrix multiplication followed by a bias offset. See the Neural Network section of the notes for more information. It is worth noting that the only difference between FC and CONV layers is that the neurons in the CONV layer are connected only to a local region in the input, and that many of the neurons in a CONV volume share parameters. However, the neurons in both layers still compute dot products, so their functional form is identical. Therefore, it turns out that it's possible to convert between FC and CONV layers.

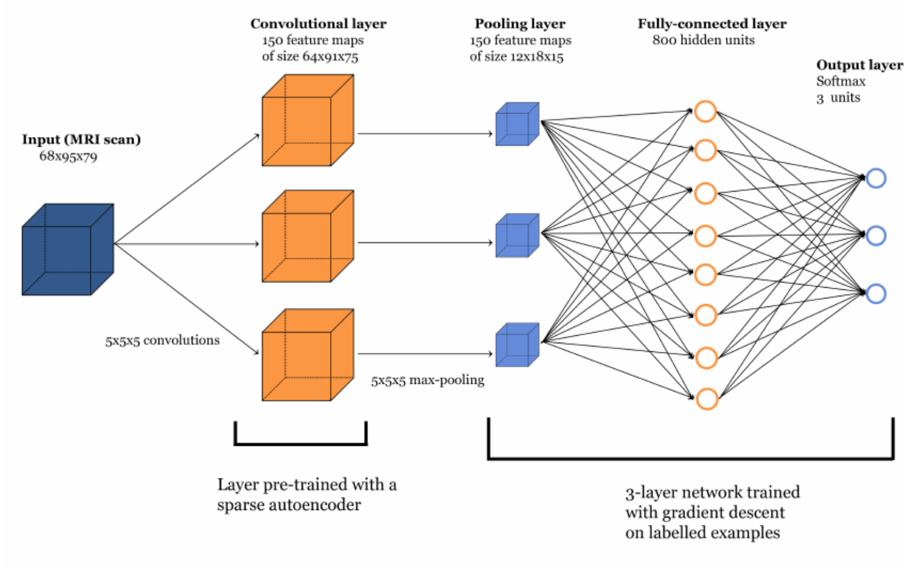


Figure 3.13: Eg. Architecture of CNN used for 3 way classification for MRI scan

3.8 Transfer Learning

Given a specific source domain $\mathcal{D}_s$ which consists of a feature space $\mathcal{X}$, class label space $\mathcal{Y}$ and a marginal probability distribution $P(X)$, where $X = \{x_1, \cdots x_n\} \in \mathcal{X}$ and $Y = \{y_1, \cdots y_n\} \in \mathcal{Y}$. For example, assuming we are to classify images into cats and dogs, then X is a learning sample and x_i is the i^{th} term vector corresponding to a class label y_i of the image either being a cat or a dog, then $\mathcal{X}$ is the space of all term vectors and the domain is given by $\mathcal{D}_s = \{\mathcal{X}, P(X)\}$. A task $\mathcal{T}_s = \{\mathcal{Y}, P(Y|X)\}$ is learned from the sample training data composed of the pair $\{x_i, y_i\}$, where $x_i \in X$ and $y_i \in Y$. An objective predictive function $f(\otimes) = P(y_i|x_i)$ predicts the class label of a new instance x, i.e $f(x)$. In the case of another different target domain denoted $\mathcal{D}_t = \{\mathcal{X}_t, P(X_t)\}$ with task $\mathcal{T}_t = \{\mathcal{Y}_t, P(Y_t|X_t)\}$, the learning of the objective predictive function $f_t(\otimes)$ in $\mathcal{D}_t$ can be improved using the knowledge in D_S and T_S , where $\mathcal{D}_S \neq \mathcal{D}_T$, or $T_S \neq T_t$. This implies that $\mathcal{X} \neq \mathcal{X}_t$ or $P(X) \neq P_t(\mathcal{X}_t)$. Adapting this knowledge from $\mathcal{T}_s$ on $\mathcal{D}_S$ to improve $\mathcal{T}_t$ on $\mathcal{D}_T$ is called transfer learning [99].

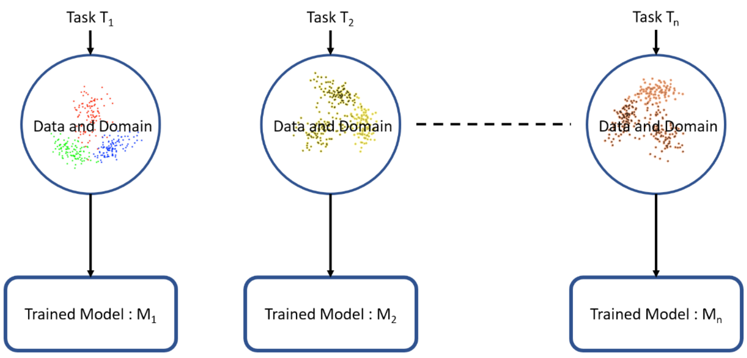


Figure 3.14: machine learning (ML) trains every model in isolation based on the specific domain, data and task.[https://www.safaribooksonline.com/library/view/hands-on-transfer-learning/]

In other words, transfer learning [19], [40] is simply adapting the knowledge that has been learned in one task eg. natural image classification task to improve the generalization performance in another related but different task such as radiology medical image classification.

Training a CNN from scratch for radiology image classification could be an extremely challenging task due to the relatively inadequate amount of labeled image data available for training and testing. This concept permits the transfer of the knowledge (representations) from a previous task trained on relatively large amount of dataset in the form of weights treated as fixed feature extractor for initializing new network architecture for small dataset.

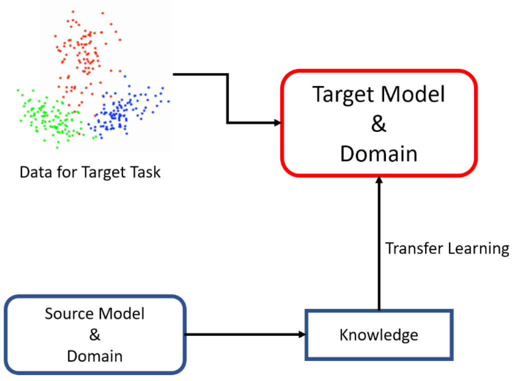


Figure 3.15: This diagram [https://www.safaribooksonline.com/library/view/hands-on-transfer-learning/] shows how transfer learning enables reusing existing knowledge for new task

Transfer learning has proven to be worth the try in deep learning anchored in the research of Bengio et al. 2014 [135] as they experimentally quantified the generality and specificity of neurons in each layer of a deep convolutional neural network. Their work concluded with the surprising result that initializing a new network with transferred features from almost any number of layers can produce a boost to generalization that lingers even after fine-tuning to the target dataset. Many machine learning competitions have found success in the architectural models built in which this concept was implemented.

Their study further revealed that the first layer of many deep neural networks trained on natural images learn features similar to Gabor filters [67], (generally used in texture analysis, edge detection, feature extraction, disparity estimation and etc) and color blobs. And these first-layer features appear not to be specific to a particular dataset or task, but general in that they are applicable to many datasets and tasks. The representations obtained could be used for other tasks irrespective of how they were obtained [54]. The entire new network could be retrained or only the final layers (mostly the fully connected layer) of the newly built network architecture is retrained on the small dataset [8].

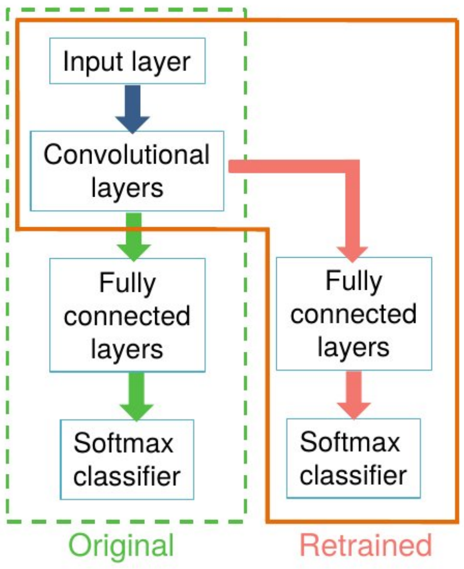


Figure 3.16: Visual concept of transfer learning applied to CNN. The rectangular green dashed architecture represents an original deep convolutional neural network trained on a large dataset for classification task. The red line indicates a new network architecture built by using the first-layers of the original architecture with a new fully connected layer built on top of it for another classification task, and then retrained [138] [https://www.safaribooksonline.com/library/view/hands-on-transfer-learning/]

3.8.1 Fine-Tuning. Fine-tuning a deep CNN is a strategy based on the concept of transfer learning. It is a proven solution that ensures the transfer of learned recognition capabilities from general domains to the specific tasks [105]. During fine tuning, the learning rates of each layer is appropriately set to relatively very low learning rates (not zero) in order not to perturb or preserve the parameters of the original or previous network from which they are been transfered into the new network or some of the earlier layers are fixed (freeze the weights of the first few layers) and only fine-tune higher-level portion of the network. The softmax classifier is set to a reasonable learning rate due to its randomized initial values.

3.9 Pre-Trained Models From ImageNet and ILSVRC

ImageNet is an image database organized according to the WordNet hierarchy (currently only the nouns), in which each node of the hierarchy is depicted by hundreds and thousands of images [39]. ImageNet is a project of the Stanford Vision Lab at Stanford University and contains over 15 million labeled high-resolution images belonging to roughly 22,000 categories [73]. The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [109] evaluates algorithms for object detection and image classification at large scale. Successful models from the challenge trained models that can correctly classify an input image into 1,000 separate object categories. Models were trained on approximately 1.2 million training images with another $50,000$ images for validation and $100,000$ images for testing. ImageNet challenge is currently the benchmark for computer vision classification algorithms [7].

3.9.1 Inception-V3 Network. GoogLeNet has a general overall independent building blocks of 100 layers. It is considered 22 layers deep CNN when counting only layers with parameters but has 27 layers when counted with pooling. It emerged the winner of ILSVRC 2014 [121] with very close to human-level performance achieving a top-5 error rate of 6.7%. GoogleNet introduced a new architectural component using a CNN called the inception layer. It is one of the first CNNs that strayed away from general approach of stacking convolutional and pooling layers on top of each other in a sequential structure. It has pieces of the network that happen in parallel. The Inception architecture adapts "Network in Network" approach by Lin et al [81] to increase the representational power of the neural networks and reduce the computational resources usage using sparsity in the layers based on the theoretical grounds given by Arora et al [15]. There are four versions of the inception module and Google provides a clearly differentiation of these here [120].

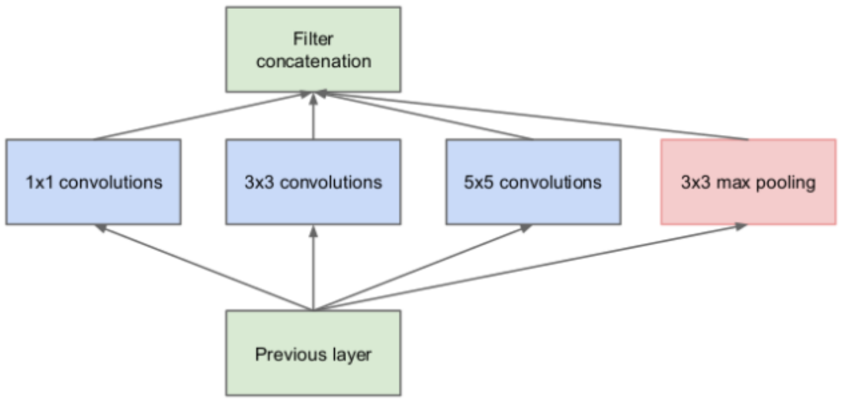


Figure 3.17: Naive Inception module used in GoogLeNet.

The Inception architecture used in ILSVRC 2014 had the following structure as denoted by Szegedy et al. [121].

- An average pooling layer with (5×5) filter size and stride 3.
- A (1×1) layer with 128 filters for dimension reduction and rectified linear activation.
- A fully connected layer with 1024 units and rectified linear activation.
- A dropout layer with 70% ratio of dropped outputs.

The inception layer convolves in parallel with different kernel sizes, starting from (1×1), (3×3) to a bigger one, such as (5×5), and the outputs are concatenated to produce the next layer. This helps to keep a fine resolution for smaller information on the images.

The inception-V3 uses the principles of factorization in the convolutions by replacing one (5×5) convolution with two (3×3) convolutions which reduces the number of parameters by 28%. A (3×1) convolution is followed by one (1×3) convolution which replaces one of the (3×3) convolutions. This further reduces the parameters by 33%. In implemetation, when n=7, two (1×7) convolutions replace two (7×7) convolutions using asymmetric factorization. See Figure 3.23. An auxiliary classifier is used

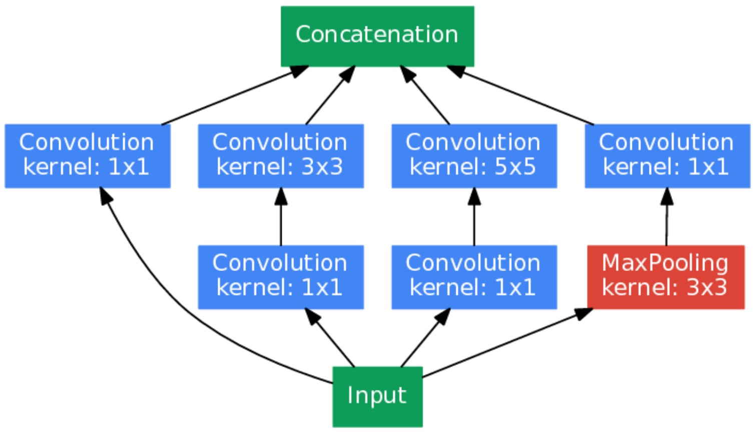


Figure 3.18: Inception module with dimension reductions

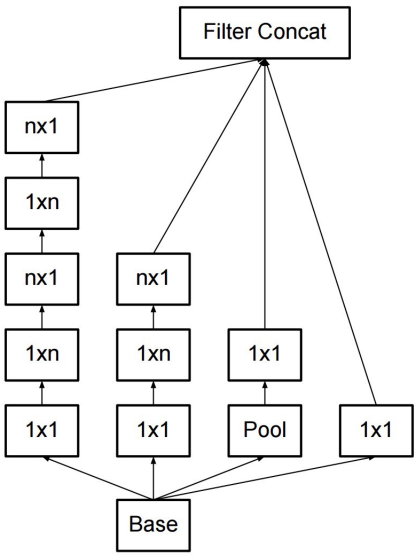


Figure 3.19: Inception modules after the factorization of the $(n \times n)$ convolutions. In our proposed architecture, we chose $n = 7$ for the (17×17) grid. Extracted from [103]

on the top of the last (17×17) layer to act as a regularization. The inception proposes a less expensive and an efficient grid size reduction instead of the conventional max pooling for feature map reduction. It uses convolutional with 2 strides to obtain 320 feature maps and again uses max pooling to obtain another 320. These two sets of feature maps are concatenated before they are sent to the next level of inception module [3].

The Inception part of the network, has 3 traditional inception modules at the (35×35) with 288 filters each. This is reduced to a (17×17) grid with 768 filters using the grid each. This is is followed by 5 instances of the factorized inception modules. This is reduced to a $(8 \times 8 \times 1280)$ grid with the grid reduction technique. At the coarsest (8×8) level, we have two Inception, with a concatenated output filter bank size of 2048 for each tile. The detailed structure of the network, including the sizes of filter banks inside the Inception modules, is shown in Figure 3.21.

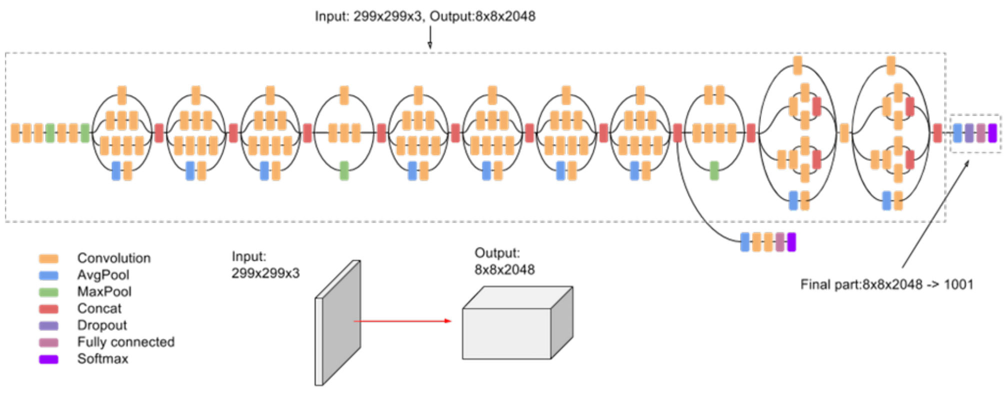


Figure 3.20: Inception module that reduces the grid-size while expanding the filter banks.

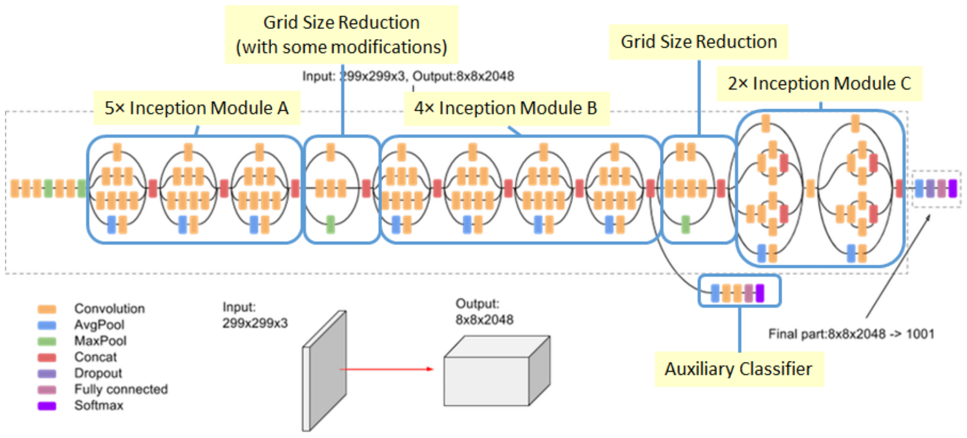


Figure 3.21: Inception module that reduces the grid-size while expanding the filter banks.

3.9.2 VGG Network. The VGG network architecture was introduced by Simonyan and Zisserman (Researchers from the Oxford Visual Geometry Group) in their 2014 paper [115]. The model achieves 92.7% top-5 test accuracy in ImageNet, which is a dataset of over 14 million images belonging to 1000 classes. It is characterized by a simple (3×3) convolutional layers stacked on top of each other in increasing depth with filters having a very small receptive field. One of the configurations utilizes 1×1 convolution filters, which is seen as a linear transformation of the input channels (followed by n on-linearity). Image inputs to the network have a fixed-size of (224×224) RGB, and the only pre-processing done to the input images is the subtraction of the mean RGB value which is computed on the training set, from each pixel.

The convolution stride is fixed to 1 pixel with a spatial padding that preserves the spatial resolution after convolution. Pooling is carried out by five max-pooling layers, which follow some of the convolution layers. Not all the convolution layers are followed by max pooling. Max pooling is performed over a (2×2) pixel window, with a stride of 2. ReLU activation is used in each of all the hidden layers. A stack of the convolutional layers is followed by three Fully-Connected (FC) layers: the first two have 4096 channels each, the third performs 1000 way ILSVRC classification and thus contains 1000 channels (one for each class). The final layer is the softmax layer.

Simonyan and Zisserman found training VGG network very challenging especically regarding convergence

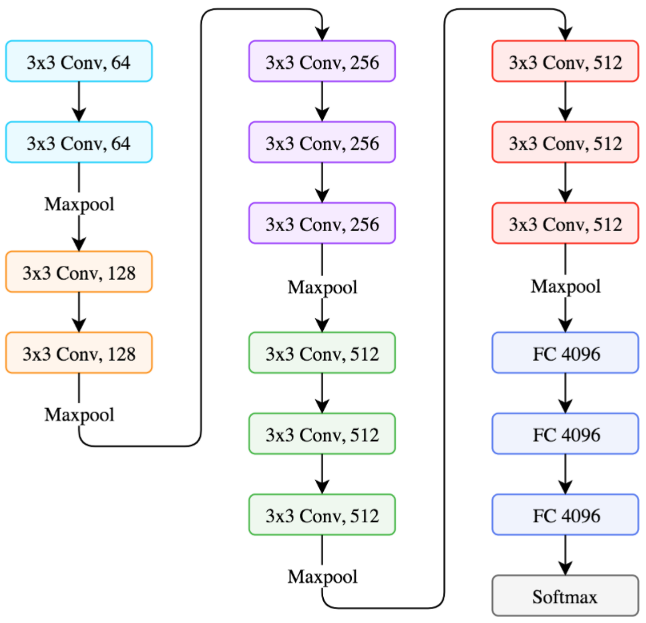
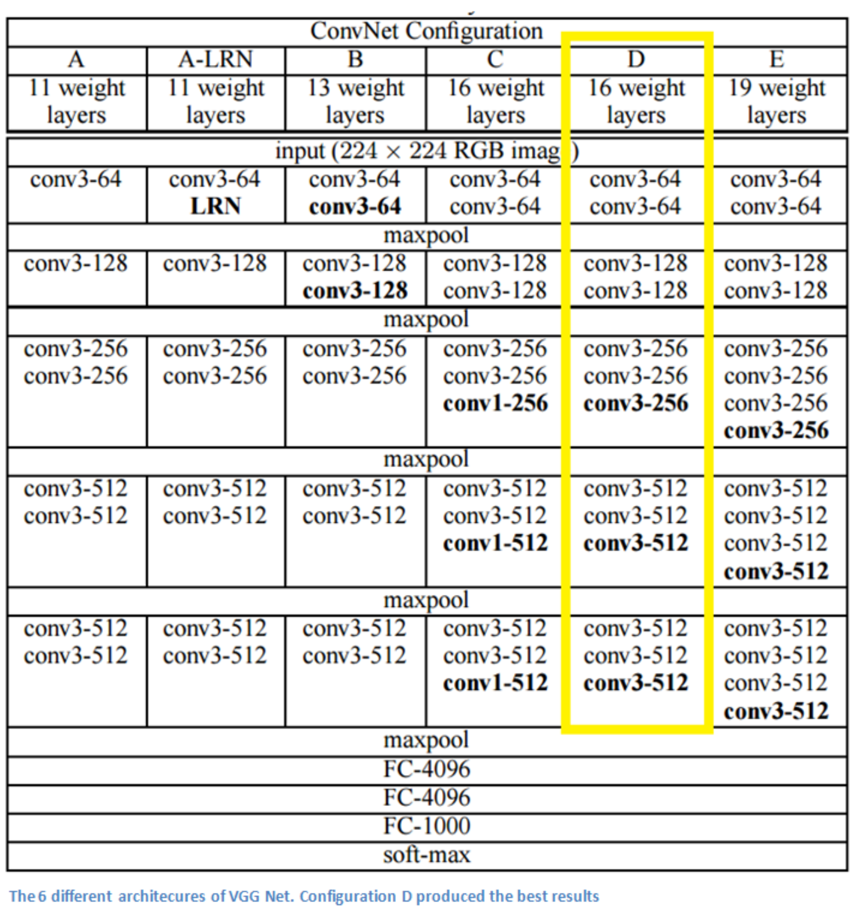


Figure 3.22: A schematic of the VGG-16 deep conolutional neural network

on the deeper networks VGG-16 and VGG-19, so they first trained smaller versions of VGG with less weight layers. The numbers "11", "13", "16" and "19" stand for the number of weight layers in the network.See Figure 3.25.



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					

The 6 different architecures of VGG Net. Configuration D produced the best results

Figure 3.23: Very Deep Convolutional Networks for Large Scale Image Recognition [115], [2]

4. Experimental Results

This chapter presents the setting of our work, data preparation and describes the analysis of our data followed by a discussion of the results. The results relate to the performance of VGG16 and Inception-V3 network classifiers. The predictive ability of these networks are demonstrated through transfer learning to classify radiology medical images which belong to eight (8) different image categories. Models are evaluated to determine the one with the best predictive ability. Hence, worth adoption as our reasoning inference engine for our clinical decision support.

4.1 Setting

Senegal is a french speaking country in West Africa in the westernmost part of a broad savannah extending across the Sahel (shore). It is located on latitude $14\,^{\circ}\mathrm{C}$ 00' N and longitude $14\,^{\circ}\mathrm{C}$ 00' W with an estimated national population of 16,479,194 as at November 2018 [9]. Senegal lies upon a low sedimentary basin characterized by an expanse of flat and undulating plains with sparse grasses and woody shrubs that experiences tropical climate [6]. There are 14 administrative regions in Senegal spanning a total land size of about 196,190 square kilometers. Dakar is its national capital.

4.1.1 Institutional Authorization. Medicis is a digital health service system developed by SEYSOO SARL (a digital service company located in Dakar, Senegal) with several integrated health care functionalities that provide clinical support to clinicians and medical staff through the collection, storage and retrieval of patient data to enhance healthcare delivery in Senegal. The basic functionalities of Medicis include:

- entry, storage and retrieval of patient's medical history, diagnoses, medications, treatment prescriptions, appointment schedules, radiology images, and laboratory test results.
- access to evidence-based tools that medical staff can use to make decisions about a patient's care.

Medicis incorperates information from other health care providers and organizations such as laboratories, authorized insurance companies, medical imaging facilities, pharmacies and emergency facilities. Hence, it contains information from the stakeholders involved in a patient's care in order to ensure that the right medical decision is made during and after diagnosis.

Ethics approval was not explicitly required since the study was granted approval and access to the private database of Medicis by the Director General of Seysoo Sarl, who couples as the supervisor of this work based on retrospective and use of anonymized medical radiology image data. The Data Protection Act (1998) and The Human Rights Act (1998) are bodies of legislation which inform our duty of patient confidentiality.

4.2 Hardware and Software Setup

New varieties of software tools currently exist which are mostly designed specifically for deep learning tasks due to the vast volume of datasets involved. Ideally, a computer with a reasonably high memory and a graphics processing unit (GPU) is required due to its high processing speed and support of CUDA®

Deep Neural Network library (cuDNN). Such computers help speed up the training of the model far many times faster than ordinary central processing unit (CPU) computers. However, a CPU computer with 8GB RAM is used in this study. The selection of the CPU is influenced by several factors which include unavailability of a GPU to carry out this task, the amount of data generated to undertake task and the language of implementation.

The Keras [32] Application Programming Interface (API) is used due its high level neural network API which runs on top of the Tensorflow [11] at the backend. The Python3 programming language is implemented in the Jupyter Notebook (http://jupyter.org/) as our chosen web application and Visual Studio Code as our IDE throughout our work.

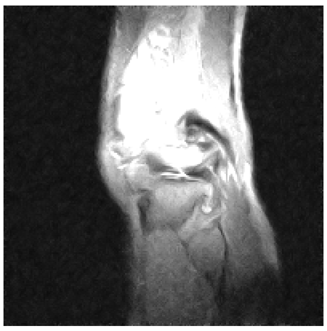
4.3 The Data Preparation

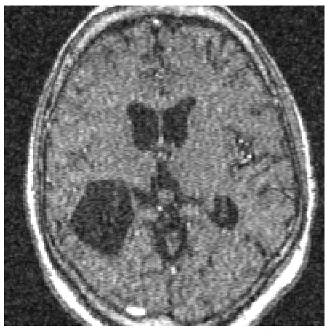
Organizational challenges and policies prolonged the data acquisiton process which in effect affected our ability to acquire our preferred viable hardware for training our models. The intent of this paper was to originally use 28,000 images which belong to 25 different categories for our models. It was however decided that we use a reduced amount of data, 7600 images, which belong to 8 different categories in order to meet the thesis deadline. This reduction came with its own advantages such as data feasibility to fit into available system memory and reduction in training time which is averaged at 48 hours training time per each model.

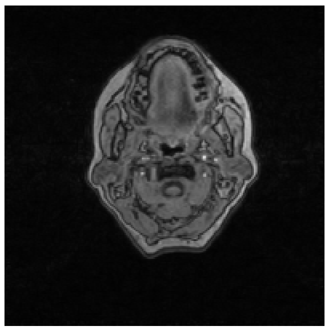
The data obtained from Medicis were of the Digital Imaging and Communications in Medicine (DICOM) format and occupied 13GB of the system's storage capacity. A DICOM image format consists of a header followed by a pixel data that stores the details about the image. The data was deidentified through the conversion process to ensure that total anonymity is achieved. The PyDicom library in Python could have been used to import the images and convert them into a numpy array for use within the Tensorflow framework, but we opted to convert the image format from DICOM to Portable Network Graphics (PNG) using the `mritopng` library which saved us much storage space ie. from 13GB to 3.5GB.

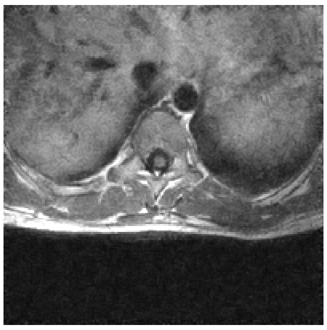
4.3.1 Image Pre-processing & Data Augmentation. Our dataset is made up of eight (8) different MRI imaging modalities which include T1 AX- weighted scans, 3DBASG AX (3D CISS), PDSAG FSAT , MRA 3D TOF, 3D T1 AX, WFS T1 AX, DWI Trace and T1 COR scans. These images were converted to PNG by using the mritopng library in Python as stated from above to ensure appropriate windowing. The 7600 MRI images were manually divided into 6000 training cases and 800 validation and test sets each. Each image category in the training set contained 500 images and the validation and training sets also contained 100 images for each category. The data were obtained between May 2017 and July 2017. Radiographs were classified into the 8 categories based on the information contained in the headers of the DICOM file as entered by the radiologist. Images were excluded if the header information were inconclusive for the unique identification of their categories.

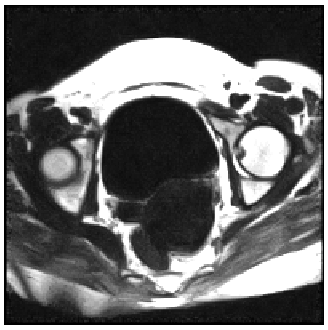
The labels for our image categories (classes) were assigned through the use of the `flow_from_directory` function of ImageDataGenerator class. This function takes the entire parent image directory as an input and assigns class indices to its sub-directories which must contain only a single class object. The ImageDataGenerator class loops the data in batches to generate batches of tensor image data with real-time data augmentation. The same data augmentation technique was implemented to augment the data size to reflect the real-world population in all our models. Images were randomly flipped horizontally and transformed through a rotation range between 0 and 20°. The width and height were shifted by

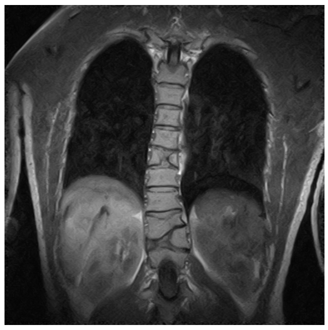

(a) PDSAGFSAT

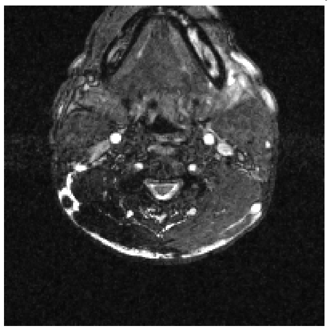

(b) MRA 3D TOF


(c) 3D T1 AX


(d) T1 AX


(e) WFS T1 AX


(f) T1 COR


(g) 3D BASG AX

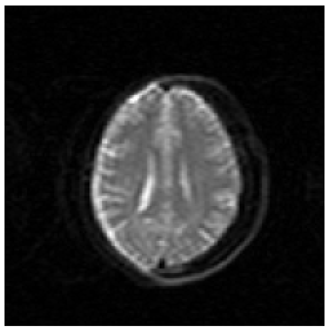

(h) DWI Trace

Figure 4.1: Sample of Medical Images in Each Category
Credit: *SeySoo Sarl*

a factor of 0.2 and the boundaries of the input images were filled using nearest mode in `fill_mode`. We rescaled all our images (grayscale) to target values between 0 and 1 by a scaling factor of $1/255$ to reduce the required storage and computation. It is worth noting that the only transformation that was done on the validation and test sets in the ImageDataGenerator class is the rescaling.

4.4 VGG16 Implementation

The VGG16 model was loaded from the Keras API without the dense layers at the top and initialized with the imagenet weights as the base model. A sequential model was built on top of the base model with a softmax output layer with 8 nodes for our image classification and a fully connected layer of 256 channels equipped with the rectification non-linearity (relu) activation function. A dropout layer was then put between the fully connected layer and the softmax layer. The output of the last convolutional layers was flattened to reshape the 3D output of the convnet to 1D. It is to be noted that, the input shape to the convolutional layers of the VGG16 architecture is fixed with a size of (224×224) with 3 channels.

We froze all layers and only chose to train the top 2 dense blocks (fully connected layer and softmax layer) of our VGG16 network. The VGG16 model is more robust to noise due to its potential to double the number of features after every maxpooling layer. The number of channels (width) of the convolutional layers start from 64 in the first layer and then doubles after each max-pooling layer until it reaches 512. The frozen VGG16 base model contained 14,714 688 non-trainable parameters, and our dense layers contained 6,424,840 trainable parameters making up a total of 21,139,528 parameters for the entire VGG16 network . For freezing the weights of a particular layer, we set this parameter to False, indicating that this layer should not be trained.

The model was compiled using the categorical crossentropy as our loss function and performance metric set to accuracy. The stochastic gradient descent (SGD) optimizer was finally implemented in our final model due to its superior performance over the adam optimizer in this work with only the learning rate set to $2e - 4$ and a decay of 0 after some series of trials. The final model was trained for 25 epochs with a specified mini-batch size of 20 for both training and validation data and saved in h5 format on the system's hard drive. The time taken to train the VGG16 classifier was 48 hours.

4.5 Inception-V3 Network Implementation

The Inception-V3 pre-trained model was also loaded from Keras API without the output layer and instantiated as the base model. All weights from the model were restored except for the final output (classification) layer. A global spatial average pooling layer was built on top of the base model with a softmax output layer with 8 nodes for our image classification task. A fully connected layer with 256 channels was equipped with the rectification non-linearity (relu) activation function. A dropout layer was also placed between the fully connected layer and the softmax layer to control overfitting by reducing the complex co-adaptations on our training data. The output of the last convolutional layers was flattened to reshape the 3D output of the convolutional layer to 1D. The input shape to the convolutional layers of the Inception-V3 architecture is also fixed with a size of (299×299) with 3 channels.

All layers were frozen but for the last batch normalization layer and last dense output layer. A total of 526,792 parameters were trained for our classification task out of a total of 22,329,384 trainable parameters. The Inception-V3 network model was compiled with categorical crossentropy as our loss function and the performance metric also set to accuracy. The stochastic gradient descent (SGD) optimizer was finally implemented with a learning rate of $2e - 4$ and a momentum of 0.9. The final model was trained for 20 epochs with a specified mini-batch size of 20 for both the training and validation data and saved in h5 format on the system's hard drive. Training also lasted for two days ie. 48 hours.

4.6 Model Evaluation

All the experiments were undertaken in two phases, the training phase and the testing phase. 21% of the sample images were equally divided into validation and test set. The test set was used in the testing phase. The remaining 79% were used to train the model, and the validation data for validating the classification model. The classifications are generally aimed at maximizing the predictive accuracy by achieving the best possible predictions of our image categories. The overall evaluation metrics are averaged across image categories having in mind the number of images (support) in each category. In our case, the categories are balanced (same number of images per category). We rely on the confusion matrices and classification reports (summarizes multiple evaluation metrics with an average computed for each image category) to have more insights into the test accuracies of our models. The best predictive model has the lowest misclassification rate which is measured in terms of proportion of misclassified cases.

We denote the total number of samples being classified by $\mathcal{N}$ and also denote misclassified sample by $\mathcal{N}_m$. Hence, misclassification or error rate is given by:

$$\mathcal{C}_e = \frac{\mathcal{N}_m}{\mathcal{N}} \tag{4.6.1}$$

Classification accuracy rate is accordingly given by:

$$\mathcal{C}_a = 1 - \mathcal{C}_e \tag{4.6.2}$$

We investigated the false positives (FP) and false negatives (FN) since they provide some details of the performance of our classifiers. A false positive occurs when an image which does not belong to a category, say 3D T1 AX, is wrongly predicted by our classifier to belong to 3D T1 AX. A false negative (FN) is the reverse of a FP. It also occurs when an image indeed belongs to 3D T1 AX, but our classifier wrongly predicts that the image belongs to an image category other than 3D T1 AX.

Given k number of image categories, we denote N_i to be the number of samples belonging to the i^{th} image category such that;

$$\mathcal{N} = N_1 + N_2 + \cdots + N_k \tag{4.6.3}$$

Let N_{ip} represent number of false positives of image category i, and N_{in} represent false negatives of image category i.

$$\alpha = \frac{N_{ip}}{N_k} \quad and \quad \beta = \frac{N_{in}}{N_k} \tag{4.6.4}$$

$1 - \alpha$ and $1 - \beta$ represent the specificity and sensitivity respectively of the i^{th} image category under consideration.

The diagonal entries in our confusion matrices represent our true positives (TP) i.e. correct classifications that were truly predicted. The vertical sums give the distribution of classes in the actual examples; the horizontal sums give the distribution of classes produced by the classifier.

$$P = \frac{TP}{TP + FP} \tag{4.6.5}$$

$$R = \frac{TP}{TP + FN} \tag{4.6.6}$$

$$F_{score} = 2 * \frac{P * R}{P + R} \tag{4.6.7}$$

The precision denoted, P, is the ability of the classifier not to label as positive an image that is negative. The recall denoted, R, is also the ability of the classifier to find all the positive samples (ie. true category for the images). The F_{score} represents the harmonic mean of our precision and recall and measures the test accuracy of our classifier. The classifier performs best when its value is close to 1 and worst when it approaches 0.

A macro-average assigns equal weight to each image category, calculates the F_{score} for each category, and then averages them to get a single number. On the other hand, a micro-average gives weight to each individual instance in each category which causes categories with larger number of instances to have more influence on the overall micro-average. Since, all the image categories have equal number of images, both the micro and macro averages are the same in the our case.

4.7 Performance - VGG16

The VGG16 classifier achieved a test accuracy of 98% in the testing phase matched against a training accuracy of 91.56% after training for 25 epochs as presented in Table 4.1. The training loss was 0.35 and a validation loss of 0.19. Figure 4.4 provides the visual details about model's performance during the training phase.

Dataset	Sample	Accuracy(%)
Training Set	6000	91.56
Validation Set	800	96.50
Testing Set	800	98.00

Table 4.1: Performance of VGG16 on image dataset

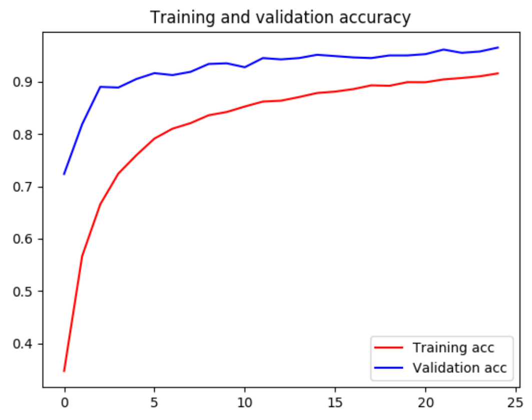


Figure 4.2: training vrs validation accuracy for VGG16 model

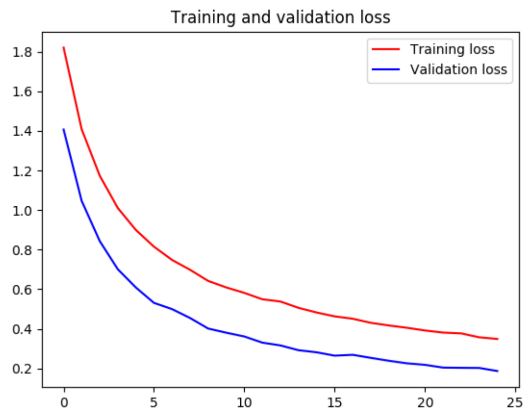
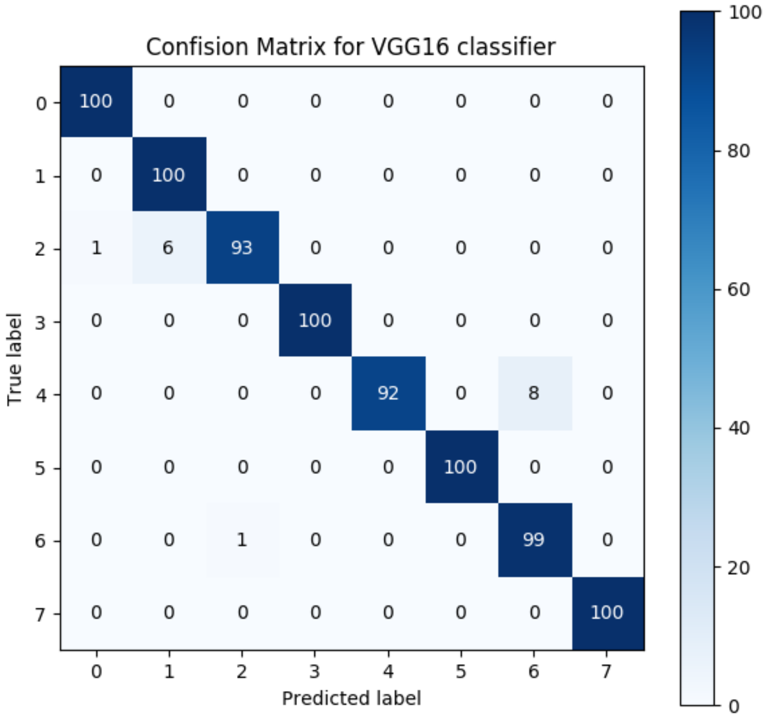


Figure 4.3: training vrs validation loss for VGG16 model

Figure 4.4: Metric Performance of VGG16 classifier

The false negatives (FN) for 3D T1 AX, 3DASG AX, MRA 3D TOF, T1 AX and WFS T1 AX were all recorded at zero (0) rates. This implies perfect classification accuracy rates for these image categories.



	0	1	2	3	4	5	6	7
0	100	0	0	0	0	0	0	0
1	0	100	0	0	0	0	0	0
2	1	6	93	0	0	0	0	0
3	0	0	0	100	0	0	0	0
4	0	0	0	0	92	0	8	0
5	0	0	0	0	0	100	0	0
6	0	0	1	0	0	0	99	0
7	0	0	0	0	0	0	0	100

Figure 4.5: confusion matrix of the eight(8) image categories and their classification results as numbers for the VGG16 classifier.

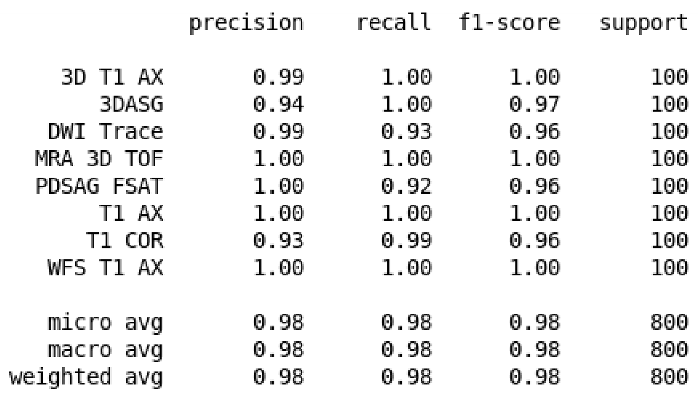
The VGG16 network model performs so well on these five (5) image categories but misses out on perfect scores for the others. DWI Trace had 7 out of its total 100 images misclassified. 6 out of the 7 were misclassified as 3DASG AX and the remaining one wrongly classified as 3D T1 AX. The PDSAG FSAT category had 8 out of its 100 total images all misclassified into the T1 COR category. The T1 COR category had only a single of its image misclassified into the DWI Trace category. Table 4.2 gives a summary of the false positive (FP) and false negative (FN) rates as obtained from the multiclass confusion matrix in Figure 4.5.

Our micro and macro averages stayed the same at 98% since there was no suspicion of class imbalance, hence, the overall weighted average of the partial accuracies of each class contributed to this percentage as observed in the classification report (Figure 4.6).

In summary, out of the total 800 images, only 16 were misclassifed. This translates into a misclassification rate of 0.02. The overall impression from the performance of the VGG16 network model indicates that the VGG16 network holds some promise for our medical image classification.

Assigned Class	Image Category	False Positive (FP) Rate(%)	False Negtive (FN) Rate(%)
Class 0	3D T1 AX	0.01	0.00
Class 1	3DASG AX	0.06	0.00
Class 2	DWI Trace	0.01	0.07
Class 3	MRA 3D TOF	0.00	0.00
Class 4	PDSAG FSAT	0.00	0.08
Class 5	T1 AX	0.00	0.00
Class 6	T1 COR	0.08	0.01
Class 7	WFS T1 AX	0.00	0.00

Table 4.2: False Positives and False Negatives of each image category for VGG16



```
              precision    recall  f1-score   support

    3D T1 AX       0.99      1.00      1.00       100
       3DASG       0.94      1.00      0.97       100
   DWI Trace       0.99      0.93      0.96       100
  MRA 3D TOF       1.00      1.00      1.00       100
  PDSAG FSAT       1.00      0.92      0.96       100
       T1 AX       1.00      1.00      1.00       100
      T1 COR       0.93      0.99      0.96       100
   WFS T1 AX       1.00      1.00      1.00       100

   micro avg       0.98      0.98      0.98       800
   macro avg       0.98      0.98      0.98       800
weighted avg       0.98      0.98      0.98       800
```

Figure 4.6: Classification report for VGG16 classifier

4.8 Performance - Inception-V3

The Inception-V3 network classifier attained a test accuracy of 97% in the testing phase matched against a training accuracy of 90.77% after training for 25 epochs also presented in Table 4.3. The training loss was 0.38 and a validation loss of 0.30. Figure 4.9 provides the graphical details about model's performance during the training phase.

Dataset	Sample	Accuracy(%)
Training Set	6000	90.77
Validation Set	800	93.00
Testing Set	800	97.00

Table 4.3: Performance of Inception-V3 on image dataset

The model achieved perfect classification scores for three image categories namely: 3D T1 AX, MRA 3D TOF and PDSAG FSAT. The false negatives (FN) recorded with this model was considerably high relative to the performance of the VGG16 network model. Two (2) images of 3DASG AX category were misclassified with an image each misclassified as T1 AX and WFS T1 AX respectively. Eight (8) images of the DWI Trace category were wrongly predicted. The model misclassified six (6) out of the 8 images to belong to 3DASG AX category and two (2) as WFS T1 AX images. Only 2 images of T1 AX category were classified as WFS T1 AX.

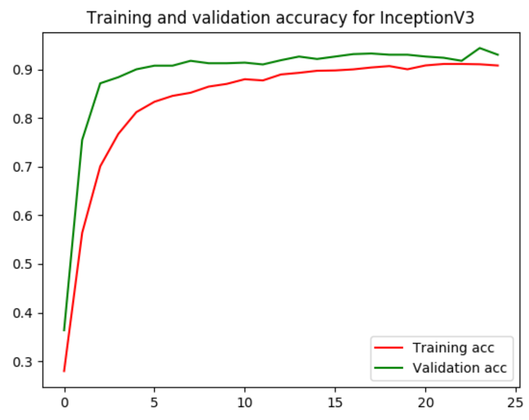


Figure 4.7: training vrs validation accuracy for InceptionV3 model

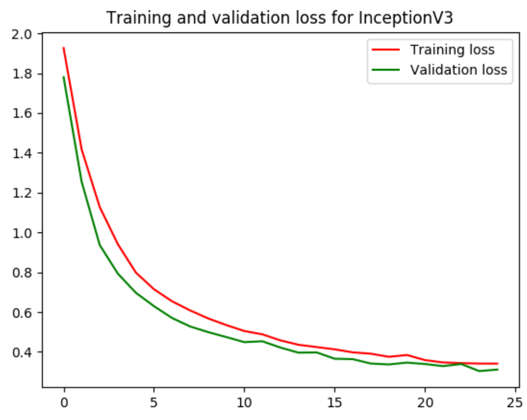
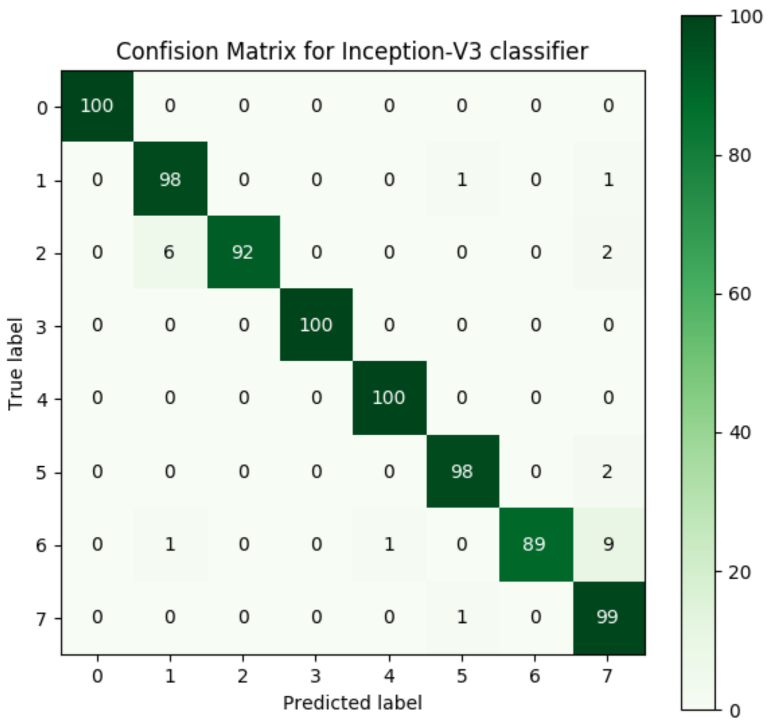


Figure 4.8: training vrs validation loss for InceptionV3 model

Figure 4.9: Metric Performance of InceptionV3 classifier



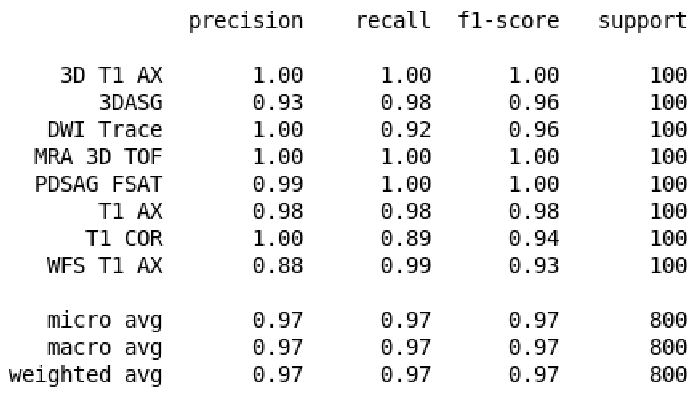
	0	1	2	3	4	5	6	7
0	100	0	0	0	0	0	0	0
1	0	98	0	0	0	1	0	1
2	0	6	92	0	0	0	0	2
3	0	0	0	100	0	0	0	0
4	0	0	0	0	100	0	0	0
5	0	0	0	0	0	98	0	2
6	0	1	0	0	1	0	89	9
7	0	0	0	0	0	1	0	99

Figure 4.10: confusion matrix of the eight(8) image categories and their classification results as numbers for the Inception-V3 classifier.

The T1 COR category had the highest number of FN. Eleven (11) of its images were misclassified with nine(9) wrongly predicted as WFS T1 AX. Single images each were also misclassified as 3DASG AX and PDSAG FSAT respectively. Only a single image of the WFS T1 AX category was misclassified as a T1 AX. It is obvious from Figure 4.5 that the VGG16 network model has learnt so well the images of the WFS T1 AX category, hence, wrongly classifying 14 images of different categories to this class. Table 4.2 summarizes the false positive and false negative performance of the model.

Assigned Class	Image Category	False Positive (FP) Rate(%)	False Negtive (FN) Rate(%)
Class 0	3D T1 AX	0.00	0.00
Class 1	3DASG AX	0.07	0.02
Class 2	DWI Trace	0.00	0.08
Class 3	MRA 3D TOF	0.00	0.00
Class 4	PDSAG FSAT	0.01	0.00
Class 5	T1 AX	0.02	0.02
Class 6	T1 COR	0.00	0.11
Class 7	WFS T1 AX	0.14	0.01

Table 4.4: False Positives and False Negatives of each image category for Inception-V3


```
              precision    recall  f1-score   support

    3D T1 AX       1.00      1.00      1.00       100
       3DASG       0.93      0.98      0.96       100
   DWI Trace       1.00      0.92      0.96       100
  MRA 3D TOF       1.00      1.00      1.00       100
  PDSAG FSAT       0.99      1.00      1.00       100
       T1 AX       0.98      0.98      0.98       100
      T1 COR       1.00      0.89      0.94       100
   WFS T1 AX       0.88      0.99      0.93       100

   micro avg       0.97      0.97      0.97       800
   macro avg       0.97      0.97      0.97       800
weighted avg       0.97      0.97      0.97       800
```

Figure 4.11: Classification report for Inception-V3 classifier

In summary, the Inception-V3 network model misclassifed 24 images. This translates into a misclassification rate of 0.03 with the F_{score} and weighted averages all at 0.97 as observed from Figure 4.11. The overall impression from the performance of the Inception-V3 network model places its performance inferior to the classification accuracy of the VGG16 model.

5. Conclusion

Findings from our experiment add up to the existing knowledge about the performance of image classifiers when they are trained on non-medical data sources such as imagenet's natural images and applied on medical target data. The results have shown that VGG16 and Inception-V3 networks both have comparable performance on accuracy and error rates. However, the VGG16 network is superior in its predictive ability over Inception-V3 in our multiclass medical image classification task. Even though the Inception-V3 network produced some good results, it is obvious that it does not perform well in identifying some modalities relative to the ability and performance of the VGG16 network. Our results conclude that fine-tuning VGG16 network produces the model with the best predictive ability and performance for our decision support. However, we are cautioned against its challenges as the techniques for taking medical radiology images improve and become more complex which may result in the need for more refined approaches to accurately detect and classify these images based on their improved modalities.

References

[1] www.aafp.org/patient-care/clinical-recommendations/cpg-manual.html.

[2] https://adeshpande3.github.io/assets/VGGNet.png.

[3] Inception-v3 -1st runner up (image classification) in ilsvrc 2015.

[4] Intrahealth international: Selected achievements. www.intrahealth.org/countries/senegal.

[5] Machine learning in healthcare: How it supports clinician decisions and why clinicians are still in charge. https://www.healthcatalyst.com/machine-learning-in-healthcare-aids-clinician-decisions.

[6] Nations encyclopedia. www.nationsencyclopedia.com/geography/Morocco-to-Slovakia/Senegal.html.

[7] 2017. www.pyimagesearch.com/2017/03/20/imagenet-vggnet-resnet-inception-xception-keras/.

[8] A. karpathy. cs231n: Convolutional neural networks for visual recognition, 2017. http://cs231n.github.io/convolutional-networks/.

[9] Senegal population, 2018. www.worldometers.info/world-population/senegal-population/.

[10] Vacuum technology & coating weblog, 2018. www.vtcmagblog.com/tag/big-data-analytics.

[11] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[12] A. Abdullah, K. S. Albeladi, and R. F. AlCattan. Clinical decision support system in healthcare industry success and risk factors. *International Journal of Computer Trends and Technology*, 11(4):188–192, 2014.

[13] N. Acır and C. Güzeliş. Automatic spike detection in eeg by a two-stage procedure based on support vector machines. *Computers in Biology and Medicine*, 34(7):561–575, 2004.

[14] A. Adehor and P. Burrell. The integrated management of health care strategies and differential diagnosis by expert system technology: a single-dimensional approach. *World Academy of Science, Engineering and Technology*, 44:533–538, 2008.

[15] S. Arora, A. Bhaskara, R. Ge, and T. Ma. Provable bounds for learning some deep representations. In *International Conference on Machine Learning*, pages 584–592, 2014.

[16] E. Balas, S. Weingarten, C. Garb, D. Blumenthal, S. Boren, and G. Brown. Improving preventive care by prompting physicians. *Journal of Lower Genital Tract Disease: Abstracts: Winter 2001*, 5(1):59–59, 2001.

[17] A. C. Barnes, A. Kpangon, P. Riley, and E. Mohebbi. Senegal private health sector assessment: Selected health products and services. 2016.

[18] R. Bellazzi and B. Zupan. Predictive data mining in clinical medicine: current issues and guidelines. *International journal of medical informatics*, 77(2):81–97, 2008.

[19] Y. Bengio. Deep learning of representations for unsupervised and transfer learning. In *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, pages 17–36, 2012.

[20] Y. Bengio, I. J. Goodfellow, and A. Courville. Deep learning. *Nature*, 521(7553):436–444, 2015.

[21] E. S. Berner, D. E. Detmer, and D. Simborg. Will the wave finally break? a brief view of the adoption of electronic medical records in the united states. *Journal of the American Medical Informatics Association*, 12(1):3–7, 2005.

[22] C. M. Bishop. Regularization and complexity control in feed-forward networks. 1995.

[23] L. Bottou. From machine learning to machine reasoning. *Machine learning*, 94(2):133–149, 2014.

[24] B. Buchanan, E. Feigenbaum, G. Sutherland, B. Heuristic DENDRAL'in Melt-zer, and D. Michie. Machine intelligence 4, 1969.

[25] J. Buckley, W. Siler, and D. Tucker. A fuzzy expert system. *Fuzzy Sets and Systems*, 20(1):1–16, 1986.

[26] J. Catlett. On changing continuous attributes into ordered discrete attributes. In *European working session on learning*, pages 164–178. Springer, 1991.

[27] B. Cestnik. Assistant 86: A knowledge-elicitation tool for sophisticated users. *Progress in Machine Learning.*, 62, 1987.

[28] V. Cheplygina. Cats or cat scans: transfer learning from natural or medical image source datasets? *arXiv preprint arXiv:1810.05444*, 2018.

[29] V. Cheplygina, M. de Bruijne, and J. P. Pluim. Not-so-supervised: a survey of semi-supervised, multi-instance, and transfer learning in medical image analysis. *arXiv preprint arXiv:1804.06353*, 2018.

[30] C.-L. Chi. Medical decision support systems based on machine learning. 2009.

[31] J. Chi, E. Walia, P. Babyn, J. Wang, G. Groot, and M. Eramian. Thyroid nodule classification in ultrasound images by fine-tuning deep convolutional neural network. *Journal of digital imaging*, 30(4):477–486, 2017.

[32] F. Chollet et al. Keras. https://keras.io, 2015.

[33] P. Clark and R. Boswell. Rule induction with cn2: Some recent improvements. In *European Working Session on Learning*, pages 151–163. Springer, 1991.

[34] D. C. Classen. Clinical decision support systems to improve clinical practice and quality of care. *Jama*, 280(15):1360–1361, 1998.

[35] P. D. Clayton, T. A. Pryor, O. B. Wigertz, and G. Hripcsak. Issues and structures for sharing medical knowledge among decision-making systems: the 1989 arden homestead retreat. In *Proceedings of the Annual Symposium on Computer Application in Medical Care*, page 116. American Medical Informatics Association, 1989.

[36] S. A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158. ACM, 1971.

[37] R. Das, I. Turkoglu, and A. Sengur. Effective diagnosis of heart disease through neural networks ensembles. *Expert systems with applications*, 36(4):7675–7680, 2009.

[38] P. A. De Clercq, J. A. Blom, H. H. Korsten, and A. Hasman. Approaches for creating computer-interpretable guidelines that facilitate decision support. *Artificial intelligence in medicine*, 31(1):1–27, 2004.

[39] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[40] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning*, pages 647–655, 2014.

[41] M. S. Donaldson, J. M. Corrigan, L. T. Kohn, et al. *To err is human: building a safer health system*, volume 6. National Academies Press, 2000.

[42] S. Dreiseitl, L. Ohno-Machado, H. Kittler, S. Vinterbo, H. Billhardt, and M. Binder. A comparison of machine learning methods for the diagnosis of pigmented skin lesions. *Journal of biomedical informatics*, 34(1):28–36, 2001.

[43] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[44] J. Durkin and J. Durkin. *Expert systems: design and development*. Macmillan New York, 1994.

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

[46] C. L. Forgy. Rete: A fast algorithm for the many pattern/many object pattern match problem. In *Readings in Artificial Intelligence and Databases*, pages 547–559. Elsevier, 1988.

[47] J. Fox, D. Glasspool, and S. Modgil. A canonical agent model for healthcare applications. *IEEE Intelligent Systems*, 21(6), 2006.

[48] J. Fox, D. Glasspool, V. Patkar, M. Austin, L. Black, M. South, D. Robertson, and C. Vincent. Delivering clinical decision support services: there is nothing as practical as a good theory. *Journal of biomedical informatics*, 43(5):831–843, 2010.

[49] J. Fox, N. Johns, and A. Rahmanzadeh. Disseminating medical knowledge: the proforma approach. *Artificial intelligence in medicine*, 14(1-2):157–182, 1998.

[50] J. Fox, V. Patkar, I. Chronakis, and R. Begent. From practice guidelines to clinical decision support: closing the loop. *Journal of the Royal Society of Medicine*, 102(11):464–473, 2009.

[51] W. J. Freeman. Three centuries of category errors in studies of the neural basis of consciousness and intentionality. *Neural Networks*, 10(7):1175–1183, 1997.

[52] A. X. Garg, N. K. Adhikari, H. McDonald, M. P. Rosas-Arellano, P. Devereaux, J. Beyene, J. Sam, and R. B. Haynes. Effects of computerized clinical decision support systems on practitioner performance and patient outcomes: a systematic review. *Jama*, 293(10):1223–1238, 2005.

[53] J. Gaschnig. Preliminary performance analysis of the prospector consultant system for mineral exploration. In *Proceedings of the 6th international joint conference on Artificial intelligence-Volume 1*, pages 308–310. Morgan Kaufmann Publishers Inc., 1979.

[54] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[55] H. Greenspan, B. Van Ginneken, and R. M. Summers. Guest editorial deep learning in medical imaging: Overview and future promise of an exciting new technique. *IEEE Transactions on Medical Imaging*, 35(5):1153–1159, 2016.

[56] C. Grosan and A. Abraham. Rule-based expert systems. In *Intelligent Systems*, pages 149–185. Springer, 2011.

[57] D. E. Heckerman, E. J. Horvitz, and B. N. Nathwani. Toward normative expert systems: Part i the pathfinder project. *Methods of information in medicine*, 31(02):90–105, 1992.

[58] R. E. Herzlinger. Why innovation in health care is so hard. *Harvard business review*, 84(5):58, 2006.

[59] D. H. Hickam, E. H. Shortliffe, M. B. Bischoff, A. C. Scott, and C. D. Jacobs. The treatment advice of a computer-based cancer chemotherapy protocol advisor. *Annals of Internal Medicine*, 103(6_Part_1):928–936, 1985.

[60] B. E. Himes, Y. Dai, I. S. Kohane, S. T. Weiss, and M. F. Ramoni. Prediction of chronic obstructive pulmonary disease (copd) in asthma patients using electronic medical records. *Journal of the American Medical Informatics Association*, 16(3):371–379, 2009.

[61] G. Hinton, N. Srivastava, and K. Swersky. Neural networks for machine learning. *Coursera, video lectures*, 264, 2012.

[62] J. L. Holt and T. E. Baker. Back propagation simulations using limited precision calculations. In *Neural Networks, 1991., IJCNN-91-Seattle International Joint Conference on*, volume 2, pages 121–126. IEEE, 1991.

[63] S. Hoo-Chang, H. R. Roth, M. Gao, L. Lu, Z. Xu, I. Nogues, J. Yao, D. Mollura, and R. M. Summers. Deep convolutional neural networks for computer-aided detection: Cnn architectures, dataset characteristics and transfer learning. *IEEE transactions on medical imaging*, 35(5):1285, 2016.

[64] K.-L. Hua, C.-H. Hsu, S. C. Hidayati, W.-H. Cheng, and Y.-J. Chen. Computer-aided classification of lung nodules on computed tomography images via deep learning technique. *OncoTargets and therapy*, 8, 2015.

[65] M. Hueso, A. Vellido, N. Montero, C. Barbieri, R. Ramos, M. Angoso, J. M. Cruzado, and A. Jonsson. Artificial intelligence for the artificial kidney: Pointers to the future of a personalized hemodialysis therapy. *Kidney Diseases*, 4(1):1–9, 2018.

[66] E. B. Hunt, J. Marin, and P. J. Stone. Experiments in induction. 1966.

[67] J. P. Jones and L. A. Palmer. An evaluation of the two-dimensional gabor filter model of simple receptive fields in cat striate cortex. *Journal of neurophysiology*, 58(6):1233–1258, 1987.

[68] A. Kandaswamy, C. S. Kumar, R. P. Ramanathan, S. Jayaraman, and N. Malmurugan. Neural classification of lung sounds using wavelet coefficients. *Computers in biology and medicine*, 34(6):523–537, 2004.

[69] R. Kaushal, K. G. Shojania, and D. W. Bates. Effects of computerized physician order entry and clinical decision support systems on medication safety: a systematic review. *Archives of internal medicine*, 163(12):1409–1416, 2003.

[70] K. Kawamoto, C. A. Houlihan, E. A. Balas, and D. F. Lobach. Improving clinical practice using clinical decision support systems: a systematic review of trials to identify features critical to success. *Bmj*, 330(7494):765, 2005.

[71] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[72] I. Kononenko and M. Kukar. Machine learning for medical diagnosis. *Lavra c, N., editor, Computer Aided Data Analysis in Medicine*, pages 9–30, 1995.

[73] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[74] E. A. Krupinski. Current perspectives in medical image perception. *Attention, Perception, & Psychophysics*, 72(5):1205–1217, 2010.

[75] A. Kumar, J. Kim, D. Lyndon, M. Fulham, and D. Feng. An ensemble of fine-tuned convolutional neural networks for medical image classification. *IEEE journal of biomedical and health informatics*, 21(1):31–40, 2017.

[76] P. Lakhani. Deep convolutional neural networks for endotracheal tube position and x-ray image classification: Challenges and opportunities. *Journal of digital imaging*, 30(4):460–468, 2017.

[77] N. Lavrač. Selected techniques for data mining in medicine. *Artificial intelligence in medicine*, 16(1):3–23, 1999.

[78] Y. Le Cun, L. Jackel, B. Boser, J. Denker, H. Graf, I. Guyon, D. Henderson, R. Howard, and W. Hubbard. Handwritten digit recognition: Applications of neural net chips and automatic learning. In *Neurocomputing*, pages 303–318. Springer, 1990.

[79] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[80] R. S. Ledley and L. B. Lusted. Reasoning foundations of medical diagnosis. *Science*, 130(3366):9–21, 1959.

[81] M. Lin, Q. Chen, and S. Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

[82] P. J. Lisboa and A. F. Taktak. The use of artificial neural networks in decision support in cancer: a systematic review. *Neural networks*, 19(4):408–415, 2006.

[83] G. Litjens, T. Kooi, B. E. Bejnordi, A. A. A. Setio, F. Ciompi, M. Ghafoorian, J. A. Van Der Laak, B. Van Ginneken, and C. I. Sánchez. A survey on deep learning in medical image analysis. *Medical image analysis*, 42:60–88, 2017.

[84] A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3, 2013.

[85] G. D. Magoulas and A. Prentza. Machine learning in medical applications. In *Advanced Course on Artificial Intelligence*, pages 300–307. Springer, 1999.

[86] M. E. Mavroforakis, H. V. Georgiou, N. Dimitropoulos, D. Cavouras, and S. Theodoridis. Mammographic masses characterization based on localized texture and dataset fractal analysis using linear, neural and support vector machine classifiers. *Artificial intelligence in medicine*, 37(2):145–162, 2006.

[87] M. A. Mazurowski, M. Buda, A. Saha, and M. R. Bashir. Deep learning in radiology: an overview of the concepts and a survey of the state of the art. *arXiv preprint arXiv:1802.08717*, 2018.

[88] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[89] A. Menegola, M. Fornaciali, R. Pires, F. V. Bittencourt, S. Avila, and E. Valle. Knowledge transfer for melanoma screening with deep learning. In *Biomedical Imaging (ISBI 2017), 2017 IEEE 14th International Symposium on*, pages 297–300. IEEE, 2017.

[90] R. A. Miller, H. E. Pople Jr, and J. D. Myers. Internist-i, an experimental computer-based diagnostic consultant for general internal medicine. *New England Journal of Medicine*, 307(8):468–476, 1982.

[91] M. Minsky and S. A. Papert. *Perceptrons: An introduction to computational geometry*. MIT press, 2017.

[92] T. M. Mitchell. Machine learning and data mining. *Communications of the ACM*, 42(11):30–36, 1999.

[93] V. Moustakis and G. Charissis. Machine learning and medical decision making. In *Proceedings of Workshop on Machine Learning in Medical Applications, Advance Course in Artificial Intelligence-ACAI99*, pages 1–19, 1999.

[94] M. Negnevitsky. *Artificial intelligence: a guide to intelligent systems*. Pearson Education, 2005.

[95] Y. Nesterov. Introductory lectures on convex optimization. applied optimization, vol. 87, 2004.

[96] N. J. Nilsson. *Learning machines: foundations of trainable pattern-classifying systems*. McGraw-Hill, 1965.

[97] A. Oguntimilehin, A. Adetunmbi, K. Olatunji, et al. A machine learning based clinical decision support system for diagnosis and treatment of typhoid fever. *A Machine Learning Based Clinical Decision Support System for Diagnosis and Treatment of Typhoid Fever*, 4(6):961–969, 2014.

[98] L. Ohno-Machado, J. H. Gennari, S. N. Murphy, N. L. Jain, S. W. Tu, D. E. Oliver, E. Pattison-Gordon, R. A. Greenes, E. H. Shortliffe, and G. O. Barnett. The guideline interchange format: a model for representing guidelines. *Journal of the American Medical Informatics Association*, 5(4):357–372, 1998.

[99] S. J. Pan, Q. Yang, et al. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

[100] J. Pearl. Bayesian networks: A model of self-activated memory for evidential reasoning. In *Proceedings of the 7th Conference of the Cognitive Science Society, 1985*, pages 329–334, 1985.

[101] J. Pearl. Probabilistic reasoning in intelligent systems: Networks of plausible reasoning, 1988.

[102] D. J. Power. A brief history of decision support systems. *DSSResources. COM, World Wide Web, http://DSSResources. COM/history/dsshistory. html, version*, 4, 2007.

[103] K. Rajalakshmi, S. C. Mohan, and S. D. Babu. Decision support system in healthcare industry. *International Journal of Computer Applications*, 26(9):42–44, 2011.

[104] L. Ramirez, N. G. Durdle, V. J. Raso, and D. L. Hill. A support vector machines classifier to assess the severity of idiopathic scoliosis from surface topography. *IEEE Transactions on Information Technology in Biomedicine*, 10(1):84–91, 2006.

[105] A. K. Reyes, J. C. Caicedo, and J. E. Camargo. Fine-tuning deep convolutional networks for plant recognition. In *CLEF (Working Notes)*, 2015.

[106] E. Ribeiro, M. Häfner, G. Wimmer, T. Tamaki, J. J. Tischendorf, S. Yoshida, S. Tanaka, and A. Uhl. Exploring texture transfer learning for colonic polyp classification via convolutional neural networks. In *2017 IEEE 14th International Symposium on Biomedical Imaging (ISBI 2017)*, pages 1044–1048. IEEE, 2017.

[107] G. Richards, V. J. Rayward-Smith, P. Sönksen, S. Carey, and C. Weng. Data mining for indicators of early mortality in a database of clinical records. *Artificial intelligence in medicine*, 22(3):215–231, 2001.

[108] F. Rosenblatt. Principles of neurodynamics. 1962.

[109] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[110] J. H. Samet, P. Friedmann, and R. Saitz. Benefits of linking primary medical care and substance abuse services: patient, provider, and societal perspectives. *Archives of internal medicine*, 161(1):85–91, 2001.

[111] J. Schreiber and P. Stern. A review of the literature on evidence-based practice in physical therapy. *Internet Journal of Allied Health Sciences and Practice*, 3(4):9, 2005.

[112] E. Shortliffe. *Computer-based medical consultations: MYCIN*, volume 2. Elsevier, 2012.

[113] E. H. Shortliffe, R. Davis, S. G. Axline, B. G. Buchanan, C. C. Green, and S. N. Cohen. Computer-based consultations in clinical therapeutics: explanation and rule acquisition capabilities of the mycin system. *Computers and biomedical research*, 8(4):303–320, 1975.

[114] P. Y. Simard, Y. A. LeCun, J. S. Denker, and B. Victorri. Transformation invariance in pattern recognition-tangent distance and tangent propagation. In *Neural networks: tricks of the trade*, pages 239–274. Springer, 1998.

[115] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[116] D. F. Sittig, A. Wright, J. A. Osheroff, B. Middleton, J. M. Teich, J. S. Ash, E. Campbell, and D. W. Bates. Grand challenges in clinical decision support. *Journal of biomedical informatics*, 41(2):387–392, 2008.

[117] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

[118] K. Srinivas, B. K. Rani, and A. Govrdhan. Applications of data mining techniques in healthcare and prediction of heart attacks. *International Journal on Computer Science and Engineering (IJCSE)*, 2(02):250–255, 2010.

[119] I. Sutskever, J. Martens, G. Dahl, and G. Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147, 2013.

[120] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, volume 4, page 12, 2017.

[121] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[122] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[123] N. Tajbakhsh, J. Y. Shin, S. R. Gurudu, R. T. Hurst, C. B. Kendall, M. B. Gotway, and J. Liang. Convolutional neural networks for medical image analysis: Full training or fine tuning? *IEEE transactions on medical imaging*, 35(5):1299–1312, 2016.

[124] N. H. Tannery, C. B. Wessel, B. A. Epstein, and C. S. Gadd. Hospital nurses' use of knowledge-based information resources. *Nursing outlook*, 55(1):15–19, 2007.

[125] S. Tunmibi, O. Adeniji, A. Aregbesola, and A. Dasylva. A rule based expert system for diagnosis of fever. *International Journal of Advanced Research*, 1(7), 2013.

[126] A. M. Turing. Computing machinery and intelligence. In *Parsing the Turing Test*, pages 23–65. Springer, 2009.

[127] D. A. Van Dyk and X.-L. Meng. The art of data augmentation. *Journal of Computational and Graphical Statistics*, 10(1):1–50, 2001.

[128] V. Vapnik. *The nature of statistical learning theory*. Springer science & business media, 2013.

[129] F. Velickovski, L. Ceccaroni, J. Roca, F. Burgos, J. B. Galdiz, N. Marina, and M. Lluch-Ariet. Clinical decision support systems (cdss) for preventive management of copd patients. *Journal of translational medicine*, 12(2):S9, 2014.

[130] X. Wang, W. Yang, J. Weinreb, J. Han, Q. Li, X. Kong, Y. Yan, Z. Ke, B. Luo, T. Liu, et al. Searching for prostate cancer by fully automated magnetic resonance imaging classification: deep learning versus non-deep learning. *Scientific reports*, 7(1):15415, 2017.

[131] D. Waterman. A guide to expert systems. 1986.

[132] A. Wright, D. W. Bates, B. Middleton, T. Hongsermeier, V. Kashyap, S. M. Thomas, and D. F. Sittig. Creating and sharing clinical decision support content with web 2.0: Issues and examples. *Journal of biomedical informatics*, 42(2):334–346, 2009.

[133] A. Wright and D. F. Sittig. A framework and model for evaluating clinical decision support architectures. *Journal of biomedical informatics*, 41(6):982–990, 2008.

[134] Y. Xue, R. Zhang, Y. Deng, K. Chen, and T. Jiang. A preliminary examination of the diagnostic value of deep learning in hip osteoarthritis. *PloS one*, 12(6):e0178992, 2017.

[135] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.

[136] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

[137] Y.-T. Zhou and R. Chellappa. Computation of optical flow using a neural network. In *IEEE International Conference on Neural Networks*, volume 1998, pages 71–78, 1988.

[138] Y. Zhu, Y. Chen, Z. Lu, S. J. Pan, G.-R. Xue, Y. Yu, and Q. Yang. Heterogeneous transfer learning for image classification. In *AAAI*, 2011.

[139] Z. Zhu, E. Albadawy, A. Saha, J. Zhang, M. R. Harowicz, and M. A. Mazurowski. Deep learning for identifying radiogenomic associations in breast cancer. *arXiv preprint arXiv:1711.11097*, 2017.

[140] Z. Zhu, M. Harowicz, J. Zhang, A. Saha, L. J. Grimm, E. S. Hwang, and M. A. Mazurowski. Deep learning analysis of breast mris for prediction of occult invasive disease in ductal carcinoma in situ. *arXiv preprint arXiv:1711.10577*, 2017.