ANKARA YILDIRIM BEYAZIT UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

EVALUATION OF FIXED RESTORATIONS ON PANORAMIC RADIOGRAPHS USING DEEP LEARNING AND AUTO-CROP

Ph.D. Thesis by

Ahmet Esad TOP

Department of Computer Engineering

August, 2023

ANKARA

EVALUATION OF FIXED RESTORATIONS ON PANORAMIC RADIOGRAPHS USING DEEP LEARNING AND AUTO-CROP

A Thesis Submitted to the

Graduate School of Natural And Applied Sciences of

Ankara Yildirim Beyazit University

In Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

in Computer Engineering, Department of Computer Engineering

by

Ahmet Esad TOP

August, 2023

ANKARA

Ph.D. THESIS EXAMINATION RESULT FORM

We have read the thesis entitled "**EVALUATION OF FIXED RESTORATIONS ON PANORAMIC RADIOGRAPHS USING DEEP LEARNING AND AUTO-CROP**" completed by **AHMET ESAD TOP** under supervision of **ASSIST. PROF. DR. MUSTAFA YENİAD** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Doctor of Philosophy.

Assist. Prof. Dr. Mustafa YENİAD	Assoc. Prof. Dr. M. Sertaç ÖZDOĞAN
Supervisor	Co-Supervisor
Assoc. Prof. Dr. Fatih NAR	Assoc. Prof. Dr. Ömer KARAL
Thesis Committee Member	Thesis Committee Member
Assoc. Prof. Dr. Yalçın İŞLER	Assoc. Prof. Dr. Yakup KUTLU
Examining Committee Member	Examining Committee Member

Prof. Dr. Sadettin ORHAN

Director

Graduate School of Natural and Applied Sciences

ETHICAL DECLARATION

I hereby declare that, in this thesis which has been prepared in accordance with the Thesis Writing Manual of Graduate School of Natural and Applied Sciences,

- All data, information and documents are obtained in the framework of academic and ethical rules,
- All information, documents and assessments are presented in accordance with scientific ethics and morals,
- All the materials that have been utilized are fully cited and referenced,
- No change has been made on the utilized materials,
- All the works presented are original,

and in any contrary case of above statements, I accept to renounce all my legal rights.

Date: _____ **Signature:** _____

Name & Surname: Ahmet Esad TOP

ACKNOWLEDGEMENTS

First and foremost, I would like to express my deepest gratitude to both my supervisor, Assist. Prof. Dr. Mustafa YENİAD, and my co-supervisor, Assoc. Prof. Dr. M. Sertaç ÖZDOĞAN for their valuable support, constructive recommendations, help, guidance, knowledge, and high motivation.

I would also like to extend my sincere thanks to Assoc. Prof. Dr. Ömer KARAL for his valuable support and guidance throughout the study.

In particular, I would like to express my heartfelt thanks to Assoc. Prof. Dr. Fatih NAR. His exceptional help, constructive recommendations, and positive guidance have played a crucial role in finding a way forward, and his continued support at the end of my thesis was invaluable.

Furthermore, I would like to express my sincere thanks to Dr. Hilal KAYA for her valuable contributions that greatly assisted the start of the study.

Finally, I want to give special thanks to my beloved wife Büşra, my wonderful son Yusuf Selim, my parents, and all my family members for their endless support, encouragement, and understanding during this challenging process.

August, 2023 **Ahmet Esad TOP**

EVALUATION OF FIXED RESTORATIONS ON PANORAMIC RADIOGRAPHS USING DEEP LEARNING AND AUTO-CROP

ABSTRACT

The objective of this dissertation is to investigate the capabilities of deep learning techniques, particularly Convolutional Neural Networks (CNNs) in dental imaging. While CNNs have been extensively used in various fields, their application in dental imaging is still limited. This thesis seeks to bridge this gap by creating a unique dataset comprising 20,973 panoramic radiographs that have been categorized into five distinct groups by three dental experts. The dataset was used to train the CNN models, including AlexNet, VGG-16, and variants of ResNet models. The models were trained using 10-fold cross-validation and data augmentation techniques to ensure robustness. The evaluation results indicated that the ResNet-101 model achieved the highest accuracy of 92.7% and the highest macro-average AUC of 0.989. Additionally, other models performed well also with accuracy scores ranging from 75.5% for AlexNet to 92.1% for Inception ResNet V2. The best result was improved to 94.5% accuracy and 0.993 macro-average AUC with the introduced auto-crop optimization that emerged from efforts to reduce the difficulty level of the dataset. These findings clearly showcase the potential of CNNs in dental imaging and pave the way for the creation of computer-aided diagnosis systems that can provide valuable auxiliary information immediately to dentists upon obtaining a patients' panoramic radiograph. Furthermore, it should be noted that the proposed dataset has significant versatility as it can be re-labeled for different problems and utilized in various studies. Thus, it represents a valuable resource for advancements in dental imaging research. Similarly, the auto-crop may be utilized for many scenarios as an end-to-end network layer. Overall, this thesis highlights the potential of deep learning techniques in dental imaging to improve diagnostic accuracy and efficiency of dental care, along with the performance gain achieved by differentiable auto-cropping alteration, and lays a strong foundation for future research on the application of these techniques.

Keywords: Artificial intelligence, computer aided diagnosis, convolutional neural networks, deep learning, dental restorations, differentiable cropping, gradient ascent, panoramic radiograph.

DERİN ÖĞRENME VE OTO-KIRPMA KULLANILARAK PANORAMİK RADYOGRAFLARDA SABİT RESTORASYONLARIN DEĞERLENDİRİLMESİ

ÖZ

Bu tezin amacı, diş görüntülemede derin öğrenme tekniklerinin, özellikle Evrişimsel Sinir Ağı (CNN) algoritmalarının yeteneklerini araştırmaktır. CNN'ler çeşitli alanlarda yaygın olarak kullanılmış olsa da, diş görüntülemedeki uygulamaları hala sınırlıdır. Bu tez, üç dişhekimi uzmanı tarafından beş ayrı gruba ayrılmış 20.973 panoramik radyograftan oluşan benzersiz bir veri seti oluşturarak bu boşluğu doldurmayı amaçlamaktadır. Veri seti, AlexNet, VGG-16 ve ResNet modellerinin çeşitleri dahil olmak üzere CNN modellerini eğitmek için kullanıldı. Modeller daha sonra sağlamlıklarını artırmak için 10 kez çapraz doğrulama ve veri artırma teknikleri kullanılarak değerlendirildi. ResNet-101 modeli en yüksek doğruluğu %92,7 ile ve en yüksek makro ortalama AUC'yi 0,989 ile elde etti. Ayrıca, AlexNet için %75,5 ile Inception ResNet V2 için %92,1 arasında değişen doğruluk oranlarıyla diğer modeller de iyi performans gösterdi. Veri setinin zorluk seviyesini düşürme çabaları sonucu ortaya çıkan oto-kırpma optimizasyonu ile en iyi sonuç %94,5 doğruluk ve 0,993 makro ortalama AUC'ye yükseltildi. Bu bulgular, diş görüntülemede CNN'lerin potansiyelini açıkça göstermekte ve hastanın panoramik radyografisini aldıktan sonra diş hekimlerine hemen değerli yardımcı bilgiler sağlayabilen bilgisayar destekli teşhis sistemlerinin oluşturulmasının önünü açmaktadır. Ayrıca, önerilen veri setinin farklı problemler için yeniden etiketlenebilmesi ve çeşitli çalışmalarda kullanılabilmesi nedeniyle önemli ölçüde çok yönlülüğe sahip olduğu belirtilmelidir. Böylece, bu çalışma dental görüntüleme araştırmalarındaki ilerlemeler için değerli bir kaynak olacaktır. Benzer şekilde, oto-kırpma, uçtan uca bir CNN'de eğitilebilir bir ağ katmanı olarak uygulanabilir ve farklı problemler için kullanılabilir. Genel olarak, bu tez, türevlenebilir oto-kırpma ile elde edilen performans kazancının yanı sıra teşhis doğruluğunu ve diş bakımının verimliliğini artırmak için diş görüntülemede derin öğrenme tekniklerinin potansiyelini vurgular ve bu tekniklerin uygulanmasına ilişkin gelecekteki araştırmalar için güçlü bir temel oluşturur.

Anahtar kelimeler: Yapay zeka, bilgisayar destekli teşhis, CNN, derin öğrenme, diş restorasyonları, türevlenebilir kırpma, gradyan yükselme, panoramik radyografi.

CONTENTS

NOMENCLATURE

Acronyms

AI	Artificial Intelligence
ANN	Artificial Neural Network
AUC	Area Under the ROC Curve
AYBU	Ankara Yıldırım Beyazıt University
CAD	Computer Aided Diagnostics
CBCT	Cone-Beam Computed Tomography
CBIR	Content-Based Image Retrieval
CNN	Convolutional Neural Network
CT	Computed Tomography
DBN	Deep Belief Networks
DCNN	Deep Convolutional Neural Network
DL	Deep Learning
DNN	Deep Neural Network
EMR	Electronic Medical Records
FC	Fully Connected
FPD	Fixed Partial Dentures
GAN	Generative Adversarial Network
GMM	Gaussian Mixture Model
GPU	Graphics Processing Unit
HMM	Hidden Markov Model
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
KNN	K-Nearest Neighbors
ML	Machine Learning
MLP	Multilayer Perceptron
NN	Neural Network
PACS	Picture Archiving Communication Systems
PCA	Principle Component Analysis
PCT	Periodontally Compromised Teeth
QLF	Quantitative Light-induced Fluorescence

R-CNN	Region-based Convolutional Neural Networks
ReLU	Rectified Linear Units
ResNet	Residual Network
RGB	Red-Green-Blue
RNN	Recurrent Neural Network
ROC	Receiver Operating Characterisctic
RoI	Region of Interest
RVG	RadioVisioGraphy
SGD	Stochastic Gradient Descent
SSD	Single-Shot Detector
STN	Spatial Transformer Network
SVM	Support Vector Machine
VRAM	Video Random Access Memory
XCeption	Extreme Inception
YOLO	You Only Look Once

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

This thesis is the revised and developed version of the published article [1].

Without direct programming, a computer may learn from data using Machine Learning (ML) [2]. Neural Networks (NNs) are one of the most frequently used methods in ML. NNs in computer science provide data-based learning. The structure and function of the human brain inspire NN.

Convolutional Neural Networks (CNNs) were first introduced in the early 1990s [3] and have since been widely used in various fields, achieving high success rates [4]. The popularity of Deep Learning (DL) has risen in recent years due to the decrease in cost and increase in processing power of Graphics Processing Units (GPUs) [5]. The breakthrough performance of AlexNet in 2012 [6] led to a renewed focus on CNNs, resulting in advanced results in visual classification [4, 7–11] in the last decade.

The performance of CNNs trained for classification can be poor when the dataset is not large enough [12]. For instance, a CNN trained with only 3840 images from the Caltech-256 dataset [13] resulted in a poor accuracy of 9.0% [4]. This is due to overfitting when the data size is small, which prevents the model from understanding the underlying structure of the problem. Deep learning architectures like CNN require a significant amount of data to be effective [14]. As a result, having a dataset with a sufficient number of examples is crucial for deep learning. This is one of the main reasons why there are only a few studies on teeth diagnosis using deep learning, as collecting, labeling, and obtaining ethical approval for a large dataset can be a difficult and time-consuming process.

Prosthetic dental treatment is a field of dentistry that aims to restore patients' oral function, comfort, aesthetics, and speech using biocompatible materials in cases of various missing teeth or oral tissue deficiencies [15]. Fixed Partial Dentures (FPD) are used to restore function, aesthetics, speech, and tissue support by treating the damage

caused by missing teeth with dental bridges and crowns [16].

Fixed dental prostheses, also known as dental bridges and crowns, are restorations that are used to replace a small number of missing teeth in the mouth. They are fabricated in a dental laboratory and attached to the remaining teeth by reshaping them. A crown (see Figure 1.1) is a type of restoration that covers a tooth with extensive material loss due to decay, fractures, or other reasons. A bridge (see Figure 1.2) is a type of restoration that replaces one or more missing teeth by attaching an artificial tooth to the adjacent teeth or dental implants and filling the gaps by taking support from the special coatings on the adjacent teeth [17].

Figure 1.1 The difference in appearance of teeth before and after crowns have been applied [18]

Figure 1.2 An example that shows the replacement of one or more missing teeth with a dental bridge [19]

When a tooth is lost for any reason, neighboring teeth tend to shift toward the gap. This shift causes gingival problems (i.e., in adjacent teeth), bone loss, deterioration of aesthetics, and changes in chewing forces. If this gap is not restored with a dental implant or bridge for a prolonged period, there may be further damage to the adjacent teeth [20].

Radiographs are used in the detection of restorations such as crowns and bridges in an intraoral examination and the diagnosis of pain and caries complaints related to these restorations. X-ray radiographs are an invaluable tool for dentists when diagnosing tooth or jaw problems. For example, while only large cavities can be diagnosed through intraoral examination, X-ray radiographs can detect early cavities [21]. In addition, X-ray radiographs can be used to identify the tooth causing an abscess, structural defects in teeth, root fractures, cysts, or tumors [22]. Furthermore, panoramic radiographs can be used to evaluate the condition of third molar teeth, the overall health of the tooth roots, and their relationship with the supporting bone [23].

During an X-ray examination, more X-rays are absorbed by denser structures (e.g., teeth and bones) than by soft tissues (i.e., cheeks and gums) before reaching the film [24]. The remaining X-rays after absorption and the unabsorbed ones reach the film, which generates an image called an X-ray radiograph (i.e., teeth appear lighter because fewer X-rays pass through the film). Dental restorations such as fillings and crowns may appear lighter or darker depending on the type of material used, but they are not radiolucent [1, 25]. Many diseases or conditions cannot be detected during a routine examination by a dentist, but X-ray imaging allows for the detection of abnormalities and pathologies [26].

Dentists often resort to the use of crown and bridge restorations to repair and restore teeth following operative and endodontic treatments [1,27]. However, these restorations are not always visible during a traditional clinical examination, making radiographs a crucial tool in their diagnosis [1, 28, 29]. The ability to accurately detect and identify restoration interfaces on radiographs is vital in preventing a variety of oral health issues, such as secondary caries, crestal bone loss, and periapical lesions [1, 30–32]. Without this capability, dental practitioners may be unable to fully assess the condition of the restoration and the surrounding tooth structure, potentially leading to further damage and the need for additional treatment [27]. Therefore, the detection and diagnosis of dental restorations using radiographs is essential because of providing comprehensive and effective dental care.

Radiographic examinations are very common in dentistry and are a vital diagnostic

tool for monitoring, tracking, and identifying oral diseases and treatment progress. Among the various types of radiography available, panoramic X-ray radiography is particularly useful due to its ability to capture a comprehensive view of the jaw and teeth in a single image [33]. This makes it an essential tool in the dentist's diagnostic armamentarium, allowing for faster treatment planning and a more accurate determination of the necessary course of treatment. With its ability to provide a detailed and comprehensive view of the oral structure, panoramic X-ray is a powerful tool in the diagnosis and management of dental and oral health issues [34, 35].

Panoramic radiographs provide a comprehensive view of a patient's entire dental structure, making them an invaluable tool in treatment planning [1, 36]. These radiographs have several advantages, including the use of low radiation, the ability to easily visualize gross anatomy and pathology, and less time and patient cooperation required [37]. The panoramic film provides a detailed image of the entire tooth structure in a shorter amount of time, and the radiation dose used is equivalent to a series of four bitewing films [38]. Due to these advantages, panoramic radiographs are widely considered the preferred type of radiograph in dentistry.

As new and emerging techniques in oral radiology continue to develop, it becomes increasingly important for dental professionals to have a thorough understanding of these techniques to provide the most effective treatment planning for fixed prostheses, implants, and maxillofacial defects [39]. While panoramic radiography is a valuable tool, it can undergo significant and unpredictable geometric distortion and has relatively low spatial resolution compared to intraoral radiographs. Additionally, patient-specific jaw curvature and positioning can result in variations in the image. Moreover, it fails to offer the low-level details that are visible in intraoral periapical radiography. Despite these limitations, it remains a vital tool in the diagnosis and management of dental and oral health issues [40].

Panoramic radiographs are challenging images to train since they also provide views of the chin, spine, and jaws in addition to the teeth, as noted in [41]. For the targeted task and the dataset, the valid Region of Interest (RoI) location may vary due to the structures included in the radiograph. One way to overcome this problem is to crop the

RoI from the images.

Using software in the diagnostic process has the potential to significantly increase the efficiency and effectiveness of healthcare services. In the medical field, the traditional approach of manually examining radiographs can be time-consuming and costly [42]. However, the use of deep learning models trained on radiographic images has proven to be a promising alternative. For example, studies have shown that the use of these models on chest x-ray radiographs obtained from pneumonia patients yielded more accurate results than traditional clinical methods [43]. Similarly, the ability to automatically extract information from panoramic radiographs can provide reliable and reproducible knowledge, assisting dentists in the diagnostic process and improving patient confidence in the accuracy of the diagnosis [44]. Software integration for the diagnostic processes is a positive development with great potential to improve the quality of patient care [45].

Deep Convolutional Neural Networks (DCNNs) have evolved and become a powerful tool in image analysis. They have been widely used with great success in Content-Based Image Retrieval (CBIR) for general imaging, and their potential has extended further into the field of medical imaging [46]. When coupled with computer-aided sensing systems, these technologies offer new opportunities for the quantitative and qualitative evaluation of stored digital medical data, specifically in radiological examination [47]. This is an important and rapidly growing research area in the field of health, as it can improve the diagnostic process by providing more accurate and efficient results. With the increasing digitization of medical data [48], the use of deep CNNs in computer-aided sensing systems can be a valuable tool for healthcare professionals in radiological diagnosis. Furthermore, in the sense of dental diagnosis, Computer-Aided Diagnostics (CAD) systems can be useful in virus pandemics, as saliva is responsible for the spread of the virus. Intraoral examination, which causes excessive salivation and gag reflex, should be limited, and intraoral radiographs should be used [49].

The use of DCNNs in the field of dental imaging holds great potential for improving diagnostic accuracy and efficiency [50]. These algorithms can quickly determine the status, location, and number of oral restorations, providing valuable initial information

to dentists during the examination process [51]. With this in mind, the focus of this thesis is on utilizing CNNs to detect the number of dental restorations present. The aim is to discover the potential of CNNs in this specific application and pave the way for the development of computer-aided diagnosis systems that can immediately provide more accurate and efficient auxiliary information about patients. Thus, the dissertation aims to contribute to the growing world of research in dental imaging by exploring the potential of CNNs in the field and ultimately improving the diagnostic process for patients.

This thesis represents an important first step toward the development of an automated classification system for oral pathologies using panoramic X-ray radiographs. To the best of my knowledge, this is the first study to utilize DL techniques for the quantitative level determination of dental restorations. Additionally, this thesis introduces a novel dataset with a large number of samples. Furthermore, the thesis presents a differentiable auto-cropping approach, which is also a novel approach, to automatically select and crop the RoI by adapting the gradient ascent optimization individually for each input image.

The goal of this thesis is to develop a model based on DL that can classify a panoramic radiograph according to the quantitative level of fixed restorations, specifically by determining the number of restorations present. To achieve this, a novel dataset and a novel auto-cropping approach have been created. Additionally, the thesis aims to evaluate the performance of different CNNs to determine the most suitable network for this dataset and task.

1.1 Literature Review

1.1.1 CAD Systems on Dental Radiography

Dental radiography has been used for a long time in clinical diagnosis, treatment, and surgery. With the advancements in technology, automated medical assistance and prediction systems have become increasingly popular as ML systems have become powerful and effective enough. In particular, DCNNs have gained significant attention in medical research and have demonstrated impressive results in the fields of radiology

and pathology [52–54]. In recent years, there has been a growing effort to develop computerized systems for the analysis of dental X-ray images [55]. This thesis also focuses on using DL techniques to classify X-ray images. Therefore, the literature review of this study mainly concentrates on the application of DL methods, particularly DCNNs, to dental radiography data.

When it comes to classifying a small dataset, it can be challenging to train a model without overfitting and still achieve good accuracy. A previous study [56] focused on evaluating the success of a treatment operation by analyzing periapical images of corresponding teeth before and after the operation. The dataset used in this study consisted of 196 patients' periapical dental radiography images, which were labeled by dental experts into three classes: 'getting better', 'getting worse', and 'no explicit change'. The authors applied a data augmentation technique, such as flipping, rotating, and changing the brightness of the image, to enrich the dataset. They also had an automated method to crop the RoIs, which are apical foreman areas, for root canal filling treatment. After the crop operation, pre-treatment and post-treatment dental RoI pairs were fed into a CNN for clinical quality evaluation of root canal filling treatment. The authors used a deep CNN architecture that was not too deep, as their dataset was small and deeper models could lead to overfitting. Their model performed better than AlexNet and much better than GoogleNet, which is much deeper than AlexNet and did not converge to their dataset. Their model achieved an F1 score of 0.749.

One of the primary challenges in the field of ML for medicine is obtaining a labeled dataset, and this is no different for studies related to dentistry. In cases of insufficient data, transfer learning can be a beneficial technique. [57] attempted to classify dental diseases using CNNs and transfer learning. The dataset used in this study included 251 RadioVisioGraphy (RVG) periapical x-ray images labeled into 3 different classes. Due to the small size of the dataset, the authors used transfer learning to improve the accuracy. The study aimed to classify dental caries, periapical infection, and periodontitis but did not include healthy teeth due to a lack of data. They trained their own model from scratch and also used VGG-16 pre-trained with ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2012 to adapt transfer learning. 180

images were used for training, 45 for validation, and 26 for testing. As they did not have a sufficient number of samples, the CNN trained from scratch did not perform well. They achieved 88.46% accuracy by applying transfer learning with VGG-16 and 73.07% accuracy by using CNN trained from scratch on 26 test samples.

Dental records play an important role in forensic identification, but the task of recording dental charts is difficult, and many dentists lack experience in this area. [58] aimed to automate the dental filing process by using dental X-ray images and a deep CNN architecture. The study extracted RoIs of a single tooth from Computed Tomography (CT) slices that consisted of 52 CT volumes, 42 of which were used for training and 10 for testing. The average number of RoIs extracted from a single tooth was 45. The study considered seven types of teeth: central incisors, lateral incisors, canines, first and second premolars, and first and second molars. The total number of RoIs obtained from the CT volumes was 35259. All axial slices, excluding the upper and lower 20%, were used as the training and test RoIs. The least number of training RoIs in 3 samplings was 11354 because the number of samples was balanced to the minimum number of RoIs in the 7 tooth types. They used data augmentation by rotating the image and transforming the intensity values of pixels. They fed the data into AlexNet, and the result was an 88.8% average classification accuracy for augmented data. Data augmentation improved the accuracy by approximately 5% for this dataset. Though the study seems to have good accuracy, they excluded the third molars from the 8th tooth type due to the small number of samples, which affects the accuracy dramatically.

The application of deep learning to dental imaging is an active area of research, with studies such as [50] exploring the use of CNN architectures for the detection and diagnosis of dental caries on periapical radiographs. The dataset in this study consisted of 3000 periapical radiographic images, of which 80% were used for training. The authors utilized a pre-trained GoogleNet Inception v3 CNN for transfer learning and achieved accuracies of 89%, 88%, and 82% for the premolar, molar, and both premolar and molar models, respectively. Even though the dataset had a sufficient number of samples, they used image augmentation techniques such as rotation, shifting, zooming, shearing, and horizontal flip to improve accuracy.

Another approach to using DL in dental imaging is to use bitewing radiographs as a dataset rather than periapical or panoramic radiographs. A study that focused on detecting dental caries [59] showed that CAD systems can perform better than humans. The authors developed a computer-aided system that detects caries and marks them on a bitewing radiograph using a bounding box. Training and testing of the system were done on a deep fully convolutional neural network (i.e., including more than 100 layers) using 3000 bitewing radiographs. They compared the performance of the model against three certified practicing dentists and found that the system outperformed in terms of recall and F1-score (i.e., 0.70).

Deep learning techniques have been widely applied in dental imaging studies, with each study focusing on different areas of dentistry. For example, [60] aimed to evaluate the effectiveness of deep CNN architectures in diagnosing and predicting Periodontally Compromised Teeth (PCT) using a periapical radiographic image dataset. The dataset consisted of 1740 images, with 60% used for training. The authors applied image augmentation techniques such as rotation, shifting, and shearing to the images. The dataset was divided into 3 classes: 'healthy tooth', 'moderate PCT', and 'severe PCT'. They used a VGG-19 architecture with a few modifications, including changing the number of Fully Connected (FC) layers from 2 to 3 and changing the number of nodes in these layers (i.e., 1024, 1024, and 512 nodes). They achieved an accuracy of 81% for premolars, 76.7% for molars, and hence 78.82% overall. However, it's important to note that the dataset was prepared by excluding some images that can lead to network uncertainty, like images of patients who have periodontal disease or are aged 12 years and younger. Also, they excluded all noisy, distorted, hazy, and partially present data. Additionally, they excluded images where a tooth has more than four roots, has undergone apical surgery or canal treatment, has moderate to severe caries, or has a full restorative crown. So, it can be said that they tried to make their data as clean as possible.

In [61], the detection of teeth that are compromised by periodontal disease from panoramic X-ray images was also studied, as in [60]. The study used panoramic radiographs as its dataset, in contrast to [60], which used periapical radiographs. The

study employed a state-of-the-art network for object detection, Faster Region-based Convolutional Neural Networks (R-CNN) for this task. Their faster R-CNN model used a pre-trained Residual Neural Network (ResNet)-101 architecture. The dataset consisted of 100 images, with 70% used for training, and was collected from patients with periodontal disease. The class labels were 'healthy tooth' and 'PCT' (i.e., included moderate and severe PCTs), where 43.7% of the whole data was PCT. The dataset was augmented by flipping images horizontally, and cross-validation was performed using the 5-fold method. The results were promising, achieving an F-measure of 0.81. Despite the limited dataset and binary-class object detection, the results were deemed satisfactory.

The classification of dental implant systems was studied in [62]. The authors used 8859 implant images to classify 11 different dental implant systems. They evaluated five different CNN models on the same dataset and found that the finely tuned VGG-16 model achieved the best accuracy. They collected 6513 panoramic X-ray radiographs and manually cropped the implant images, resulting in a total of 8859 implant images. Their simple CNN with 3 convolutional layers achieved an accuracy of 86%, while the finely tuned VGG-16 model achieved 93.5% classification accuracy, which was the highest among the 5 models. The authors claimed that this system may help determine dental implant brands from panoramic radiographs in the future.

Some studies in the field of dental imaging are focused on developing new techniques rather than being the first to tackle a specific problem. For example, [63] aimed to classify teeth into four categories: incisors, canines, premolars, and molars. Utilizing 400 images obtained from Cone-Beam Computed Tomography (CBCT), the authors applied data augmentation techniques and employed a 7-layer deep CNN, achieving an overall accuracy of 87%. They also evaluated the performance of max-pooling and average-pooling and found that max-pooling yielded better results. Compared to other state-of-the-art techniques at the time, their results were superior.

Other studies have employed object detection or segmentation techniques (i.e., the process of splitting an image into groups of pixels [64, 65]) rather than simply classifying teeth on radiograph images, as seen in [61]. In [41], a novel approach to

teeth segmentation on panoramic radiographs was proposed using DL techniques. This study highlighted the challenges of using panoramic images, which often include images of the chin, spine, and jaws in addition to the teeth, so they are much harder to train. The majority of earlier studies in this subject, as was already indicated in [41], have primarily relied on intra-oral radiographs, such as bitewing or periapical images, with relatively few studies utilizing panoramic radiographs [66].

In [67], a deep learning-based algorithm for teeth detection and numbering from periapical films was proposed. The approach uses a combination of three post-processing techniques, including filtering overlapping boxes identified by Faster R-CNN, detecting missing teeth with a Deep Neural Network (DNN), and a rule-based algorithm that matches the labels of detected teeth boxes with a template for correction. The study used 1250 periapical films, with 800 of them used as the training dataset for R-CNN and DNN. The results showed that the algorithm performed similarly to a junior dentist, with precision and recall rates exceeding 90%. In [68], the study also tackled the task of teeth detection and numbering, but this time using panoramic radiographs. The study used 1352 images for training and 222 images for testing, which were also augmented. The teeth detection architecture was based on Faster R-CNN, while the numbering module employed a VGG-16 architecture with a heuristic algorithm. The system's performance in terms of sensitivity was found to be 0.39% lower than those obtained by dental experts, while the teeth numbering result was 0.93% lower. In [69], they used 3D sparse voxel octrees (i.e., 3D dental models) and CNNs rather than radiographic images to train a model that segments and classifies teeth. The study achieved a segmentation accuracy of 89.81%, which was the state-of-the-art result in the field at the time. Another study, [70], focused on CT image segmentation, specifically segmenting mandibular canals in the lower jaws by applying CNNs to 637 cone-beam CT volumes. The results were accurate and demonstrated the potential for DL methods to reduce the workload of medical experts. Lastly, [71] proposed a method for detecting periodontal bone loss using panoramic radiographs, which also provides the corresponding teeth number for detected lesions. Their method, called DeNTNet, uses CNNs and transfer learning techniques and was trained on a dataset of 12179 radiographs, which were augmented.

The system achieved an F1 score of 0.75, while the average performance of the five dental clinicians in the study was 0.69.

Other than radiographic images, [72] used a different type of dental image, called Quantitative Light-induced Fluorescence (QLF) images, to automatically classify dental red autofluorescence plaque. They had a dataset of 427 QLF images, which were transformed into Red-Green-Blue (RGB) channel images and labeled according to the degree of red fluorescent plaque accumulation. 80% of the dataset was used for training, and they tested several classifiers for this task. They found that the best model was a CNN, which achieved an average F1 score of 0.75.

There are only a few studies that focus on dental restorations, but they are not quite the same as the focus of this thesis, except for one (i.e., the publication from the thesis [1]).

In [73], a study was conducted to automatically detect and classify dental restorations using panoramic radiographs. The dataset consisted of 83 X-ray images containing a total of 738 dental restorations, which were labeled into 11 different types. The dental restorations were automatically cropped and segmented, and a gray-level thresholding method was used for detection and a Support Vector Machine (SVM) for classification. The system had an accuracy of 90.5% in correctly identifying (i.e., detecting) the restorations. On the other hand, [74] aimed to classify amalgam, composite resin, and metal-ceramic restorations using CNNs. Using the ResNet-34 architecture, they were able to classify amalgam, composite, and metal-ceramic restorations with an accuracy of 88%, 87%, and 96%, respectively, using 550 bitewing and periapical radiographs.

While these two studies [73, 74] focused on identifying the specific types of restorations present in dental radiographs (i.e., panoramic in [73], bitewing and periapical in [74]), the aim of this thesis is to classify the quantity level of restorations present on panoramic radiographs. Instead of performing object detection, this study aims to classify panoramic X-ray radiographs according to the level of dental restorations in a person's mouth.

The most similar study to this thesis is found in [51]. They focused on detecting dental

restorations, dentures, or implants from panoramic radiographs using various object detection methods such as R-CNN, Faster R-CNN, Single-Shot Detector (SSD), and You Only Look Once (YOLO) v3. They employed various neural network architectures, such as ResNet-50, ResNet-101, Extreme Inception (XCeption) 101, VGG16, and DarkNet53, as the backbone and feature extractor. A total of 684 objects in 123 panoramic radiographs were fed and trained to detect one of three classes. The best detection performance was obtained with the Faster R-CNN RegnetX method, with an mAP score of 0.973, an AR score of 0.771, and an area under the precision-recall curve score of 0.952.

In contrast to [51], which focuses on object detection of dental restorations, dentures, or implants from panoramic radiographs, the study in [1] (i.e., publication from this thesis) focused on determining the quantitative level of dental restorations from these images. 20,973 panoramic X-ray radiographs were used, and they were labeled into five distinct categories. AlexNet, VGG-16, and variants of ResNet models were trained, and the best result was achieved with ResNet-101, with an accuracy of 92.7% and a macro-average Area Under the ROC Curve (AUC) of 0.989. Now, that result has been improved by proposing the auto-cropping algorithm in this thesis.

1.1.2 Differentiable Image Cropping

Auto-cropping is a crucial task in computer vision applications [75], enabling the automated extraction of relevant regions from images. The trouble of image cropping or downsampling after localizing in an end-to-end manner has been tackled in several ways [76–82].

Image transformations and cropping were generally held on two-stage training by using Spatial Transformer Networks (STNs) [82], but there are also some end-to-end approaches like [76], which also used a two-staged method. The first stage includes detecting vertebrae and segmenting the whole spine, while the second stage accurately predicts the bending directions of the localized vertebrae (i.e., a regression network). An inter-stage transfer method (i.e., differentiable cropping) is designed to deeply couple both stages. The proposed method is task-specific, connects two stages, and

localizes many bounding boxes inside one image to crop and send to the regression network. Also, the training in this study needs supervised learning (i.e., ground truth bounding boxes), but the method proposed in this thesis does not. Furthermore, to make the cropping operation differentiable, they used the RoI Align pooling layer from [83], which does not touch the raw input but gets a combination of results from localization and segmentation networks.

CNNs often integrate numerous downsampling operators, such as strided convolutions or pooling layers, that progressively decrease the resolution of intermediate features, offering some shift-invariance in addition to minimizing computational complexity [77]. The study [77] proposed the first downsampling layer with learnable (i.e., differentiable) strides by using both spatial and frequency domains similar to [78]. Unlike [78], their approach uses backpropagation to learn the size of its cropping box rather than optimizing a fixed-sized bounding box whose location is updated by a stride parameter (similarly, the approach introduced in this thesis learns both the size and the location of the bounding box). Similarly, one of the pioneering works in differentiable semantic segmentation is the research conducted by Dai et al. [79]. They introduced a differentiable pooling layer called "RoI Warping Layer," which downsamples the feature maps taken from the previous layer and interpolates them for the next layer's resolution. However, approaches in all 3 studies [77–79] were for intermediate layers to downsample rather than work on the raw image input, which is the sense of the proposed method.

The study that focuses on a task-specific supervised differentiable RoI transform layer that detects the object (i.e., the person) from the input was proposed by Han et al. [80]. However, this approach needs to perform supervised learning and was created for the object detection task. As stated in the study, the detector was separately trained in the state-of-the-art methods [83–86], and supervision was once again needed to detect the bounding boxes, which illustrates the starting point of this thesis as there is no supervision needed in the proposed method.

The idea of end-to-end auto-cropping proposed in this thesis is very similar to the idea of end-to-end applicability presented in [81]. They proposed a saliency-based warping

layer for NNs where their distortion held non-uniformly. However, the study is not the same as the proposed method because the auto-crop in this thesis is downsampling the input uniformly.

[82] introduced the concept of spatial transformers, a learnable module that enables explicit spatial manipulation of data within CNNs. The spatial transformer module allows for dynamic transformations of feature maps, such as translation, scale, rotation, and non-rigid deformations. While both the spatial transformers and the proposed auto-cropping algorithm include spatial manipulation in CNNs, the proposed auto-cropping algorithm was designed for RoI selection and cropping in images rather than achieving general spatial invariance. The proposed method in this thesis also employs a differentiable pre-processing step that includes max pooling and min pooling difference to make edges detectable and lower the resolution.

The spatial transformer module relies on bilinear interpolation. One significant disadvantage of bilinear interpolation is that it is highly localized (i.e., considers only the four nearest pixel neighbors of the query) and the gradients only flow through the intensity difference between nearby pixels in the input image, as mentioned in [87]. [87] attempted to mitigate this weakness by making the sampling operation more linear (i.e., by generating random auxiliary sample locations, then performing bilinear sampling and achieving a more linear approximation). However, even though they provided a larger context to the local transformation, the optimization was still not affected by all pixels. The proposed auto-cropping optimization in this thesis was affected by all pixels (i.e., in all cases, although some pixels affect the optimization close to zero, they are never zero, as in the two related studies).

1.2 Thesis Organization

This thesis is composed of four main chapters. Chapter 1 provides an overview of the purpose and scope of the thesis and a summary of the related works. Chapter 2 covers background information on the materials and methods utilized in this study. Chapter 3 goes into detail about the implementation and dataset used in the study. Finally the results are presented along with a discussion of the findings and potential

future directions in Chapter 4.

CHAPTER 2

MATERIALS AND METHODS

2.1 Machine Learning

"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E." [88] is the formal definition of machine learning.

Humans collect information through seeing cases or experiencing things, as learning through experiences is something that both people and animals do intuitively. When a new case occurs where the result is unknown, the collected knowledge is applied to that case. ML enables machines to learn from previous experiences; ML techniques learn from the data itself without the need for any other information. ML techniques don't use any specifically programmed selections to decide the answer; they essentially build the path to the answer using just the data.

Patterns or regularities in data are discovered by ML; hence, it can make judgments, estimations, or take actions. The larger the size of the data, the higher the performance of ML; hence, the size of the data is vital for ML. A simplified overview of the ML workflow is illustrated in Figure 2.1.

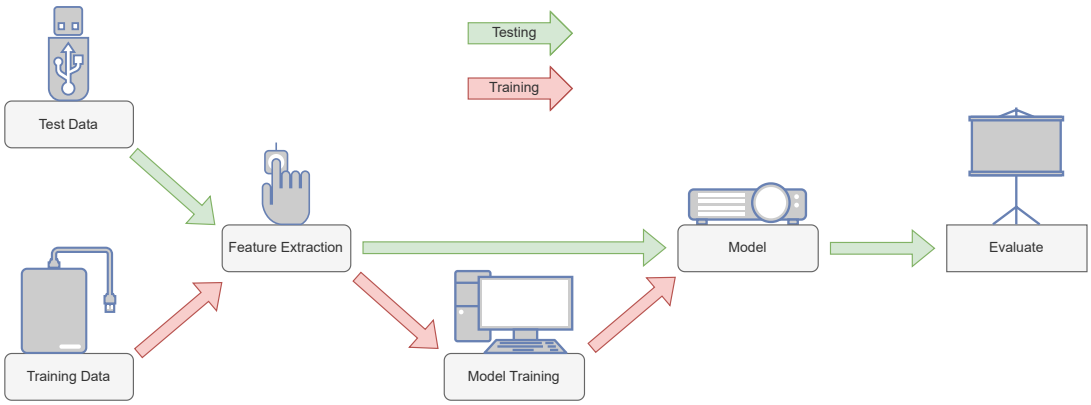


Figure 2.1 A simplified overview of a machine learning workflow. The machine learning model is trained with 'training data', whose collection is huge. Once it is trained, it can be applied to make predictions for another input data (i.e., test data)

When developing explicit algorithms is challenging or those algorithms fail, ML is

employed (i.e., in a variety of applications). Speech recognition, bioinformatics, and computer vision are some exemplary application areas of ML.

Both supervised and unsupervised learning can be considered as the general distinguishing categories for ML methods, as can be seen in Figure 2.2. This dissertation involves supervised learning, as the data is labeled by supervisors.

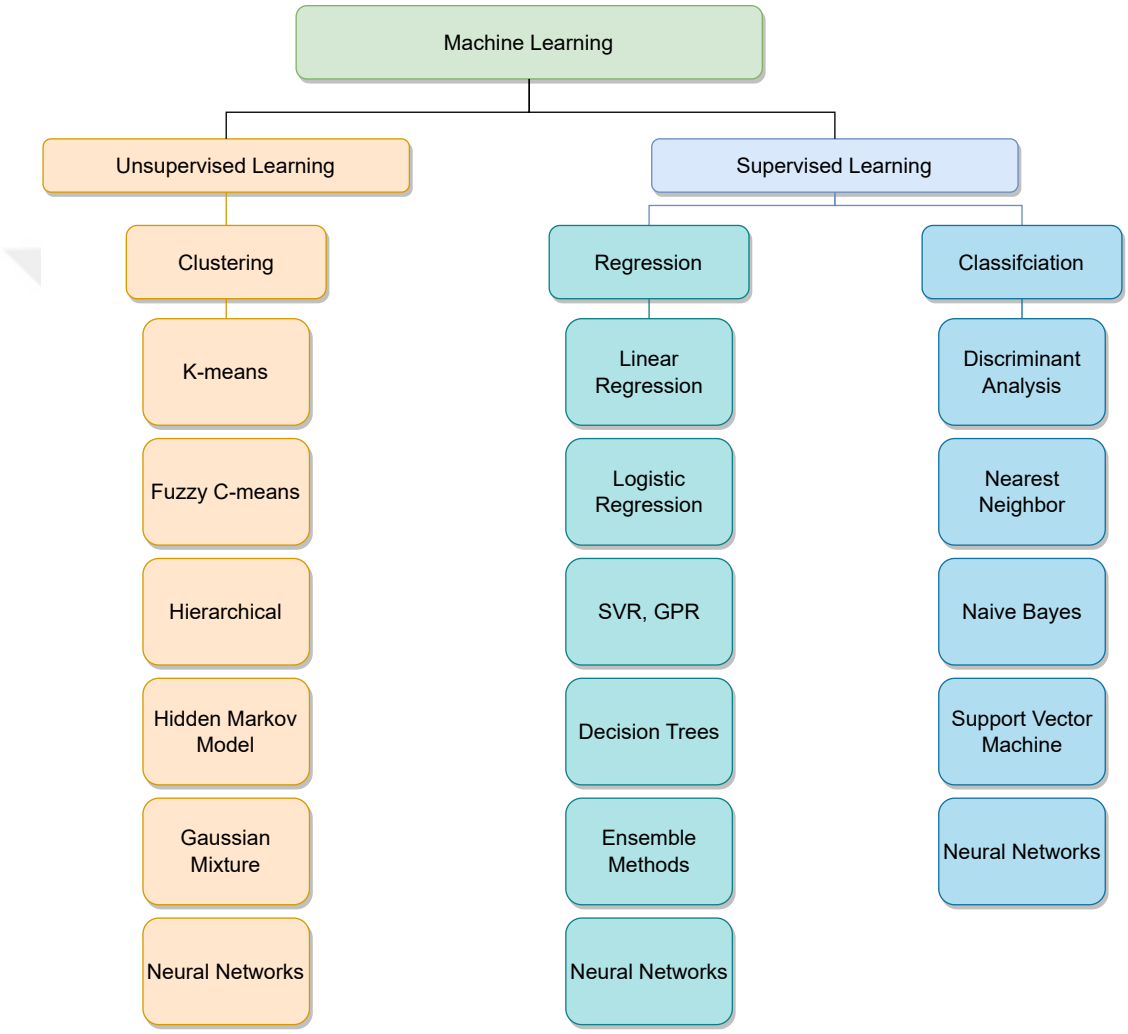


Figure 2.2 ML methods for different learning types

The input data in unsupervised learning has no labels, but the methods can find some patterns and relations without the corresponding output (see Figure 2.3). To model the data, it learns the data's fundamental structure and distribution, where there is neither a supervisor nor a correct answer. The data may be categorized after the fundamental structure (i.e., similarities or differences) has been revealed. K-Means clustering, K-Nearest Neighbors (KNN), Apriori algorithm, Principal Component Analysis (PCA),

and Hidden Markov Model (HMM) are examples of popular unsupervised learning algorithms. Supervised learning, on the other hand, is a completely different story, as the data has labels.

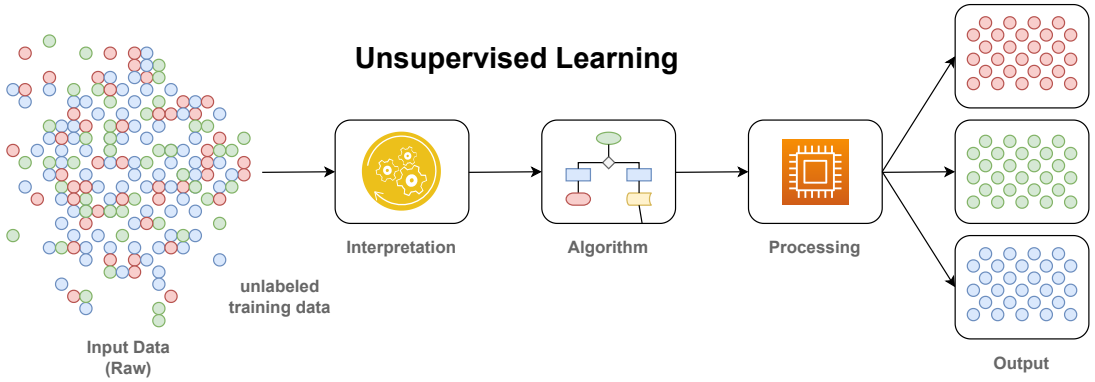


Figure 2.3 A visual representation of the concept of unsupervised learning, where the algorithm learns to identify patterns and relationships in data without the need for explicit labels or supervision

2.1.1 Supervised Learning

For supervised learning, data must be composed of pairs of input subjects and desired outputs. It learns a rule from sample inputs and their corresponding outputs from labeled data. It generates a function that can find unknown outcomes for new inputs after training with a sufficient number of examples. The intended outcome is that it can accurately discover the labels of new inputs, where success depends on the generalization capacity of a learning algorithm.

Figure 2.4 depicts an example of a supervised learning flow in which a supervisor assigns labels to data, which is then processed by one of the supervised learning algorithms to generate the desired function.

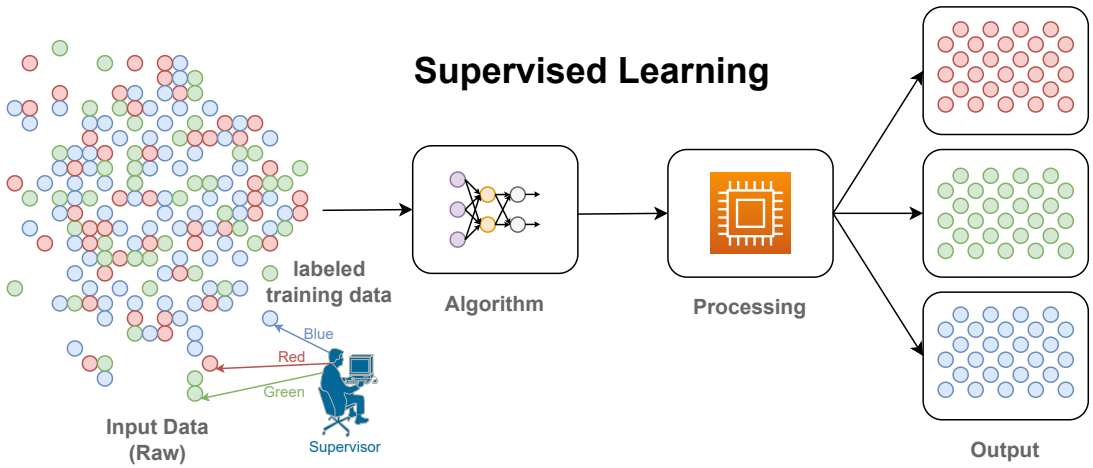


Figure 2.4 A visual representation of supervised learning, where the algorithm is trained on a labeled dataset and uses the labeled examples to make predictions on new, unseen data

Regression and classification are the two main methods employed in supervised learning, where they are used to generate predictive models. Since the outputs of the regression algorithms are real values, such as "temperature", they are suitable for estimating continuous outcomes. Since the outputs of classification systems are like "yes" or "no", they are suitable for predicting discrete outcomes. Naïve Bayes, KNN, Support Vector Machine, decision trees, Artificial Neural Networks (ANN), and logistic regression are examples of popular supervised learning algorithms. The supervised learning approach was applied in this dissertation.

2.2 Artificial Neural Networks

One of the most popular ML approaches is artificial neural networks, also referred to as neural networks. It is an information processing system, and it can be employed for unsupervised or supervised learning. The structure and function of the human brain (i.e., biological neural networks) serve as an inspiration for ANN.

ANNs learn from examples of how certain jobs will lead to certain results and what needs to be done to reach those results. There is no pre-knowledge or set of programmed rules for the task expected to be performed by the ANN before the training. An ANN simply takes input data (i.e., example data) and learns the ability to perform the required task by parsing the data and detecting patterns inside the data.

While an ANN learns how to perform a certain task (e.g., categorizing animals from

images like ‘wolf’, ’giraffe’, or ‘dog’) by using sample images, it gets the image with a corresponding output (i.e., labels) as its training data. Labels for the input data should be arranged by a supervisor, and they should be in the form of described pairs. This is the supervised learning approach.

ANN, for example, uses the first sample image as its input (i.e., when training started initially), feeds forward, and then receives the output of the first image when categorizing animals from images. According to the output, it measures the error and analyzes how close it is to the intended result. After that, it makes some adjustments to the weights by using the gradient descent algorithm to achieve the intended result. Its weights are more accurately adjusted after several iterations using various samples to reach the desired result.

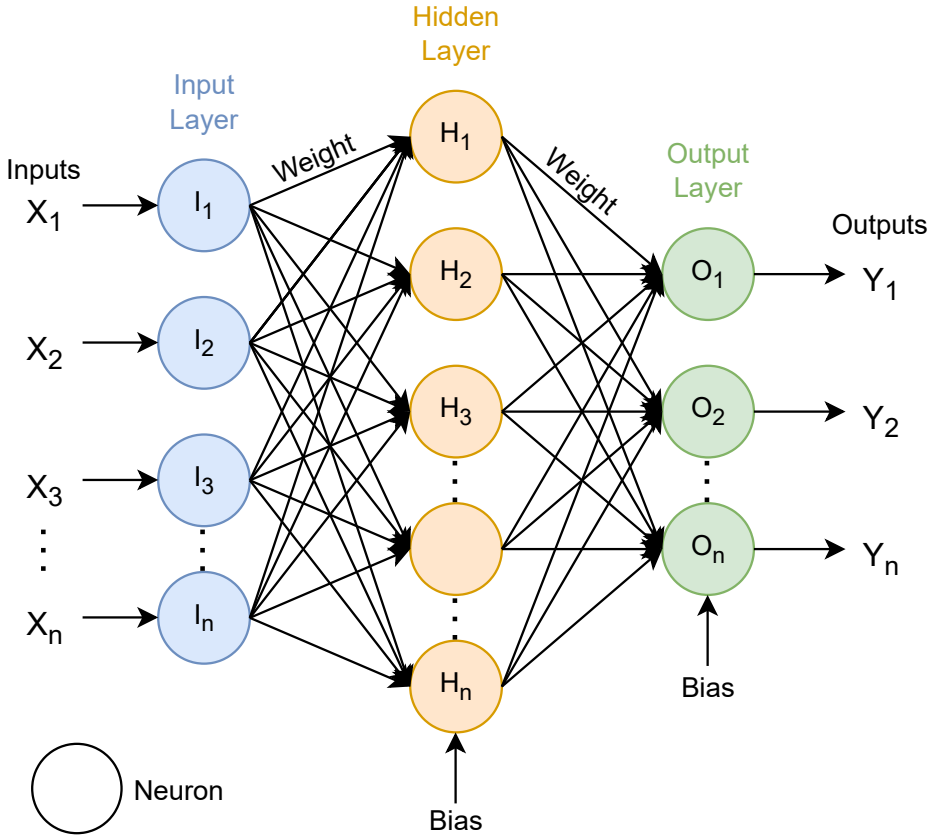


Figure 2.5 An illustration of the architecture of an ANN, showing the various components and their interconnections that allow the network to process and analyze data.

An ANN is a network made up of several nodes, where each node communicates with linked nodes by transmitting the necessary data. Once the received data has been processed by the receiving node, the processed data is sent to other linked nodes.

These networks might include nodes that are fully or partially connected (e.g., see a fully connected illustration of an ANN in Figure 2.5).

Edges, which are like neurotransmitters, are the connections between nodes (i.e., nodes are referred to as "artificial neurons" and are comparable to biological neurons). Each edge has its own weight, which is going to be updated after a backpropagation pass. Each node uses a nonlinear function and produces its result as an output, where the function's input is the sum of all the inputs of the node. Layers are groups of neurons that are on the same level, and for different layers (i.e., different levels), calculations may be different. Each layer waits until the preceding layer has completed all of its computations before beginning its own.

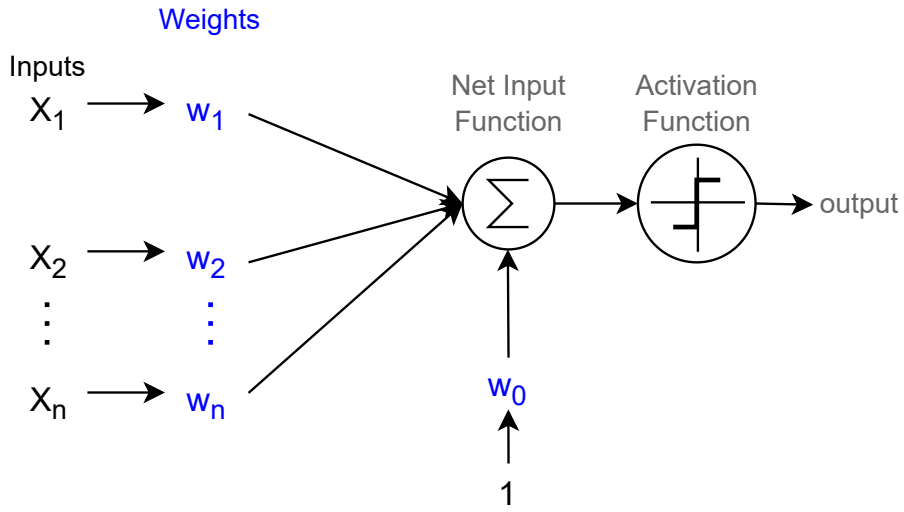


Figure 2.6 A depiction of Rosenblatt's Perceptron, which is an early model of a single-layer ANN

In 1943, the first artificial neuron was developed [89], but the technological level at the time was inadequate for much. Then, in 1958, Rosenblatt [90] invented the perceptron (Figure 2.6), but his single-layer perceptron was unable to solve the XOR issue until the backpropagation method [91] was created in 1975 (i.e., it successfully solved the XOR issue).

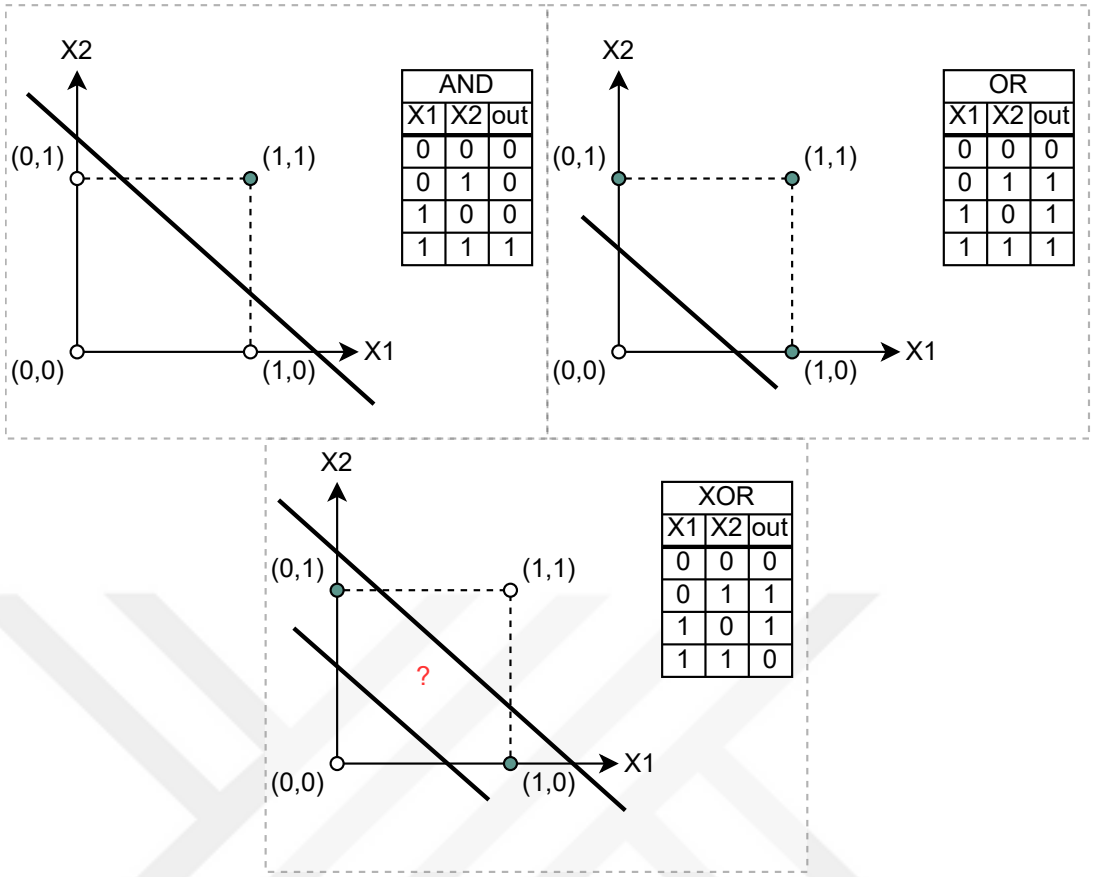
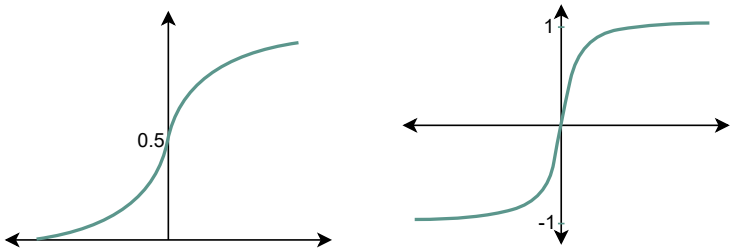


Figure 2.7 A diagram that shows the AND, OR, and XOR logic gates plotted on a Cartesian coordinate system, with each gate being distinguished by a single line that separates them

One-layered perceptrons (excluding the input layer) can be used to solve "AND" and "OR" gates, but a one-layered perceptron cannot be used to create an "XOR" gate as a single line is not enough to split "XOR" in a Cartesian plane. "XOR" must be separated by using at least two lines, as can be seen in Figure 2.7. As a result, at least a two-layered network (excluding the input layer) is required (i.e., when the input layer is included, a three-layered perceptron). This means that at least two hidden layers are needed, as one or more layers between the input and the output are known as hidden layers (i.e., their units are known as hidden units), and the term "Multi-layer Perceptron (MLP)" refers to this formation. Every node except the input layer uses a nonlinear activation function for its output (i.e., see the two most popular examples in Figure 2.8).



Sigmoid

$$\sigma(\Sigma) = \frac{1}{1 + e^{-\Sigma}}$$

Hyperbolic Tangent

$$tanh(\Sigma) = \frac{e^{\Sigma} - e^{-\Sigma}}{e^{\Sigma} + e^{-\Sigma}}$$

Figure 2.8 Two graphs illustrate the sigmoid function and the hyperbolic tangent function, which are both commonly used in ANN as activation functions

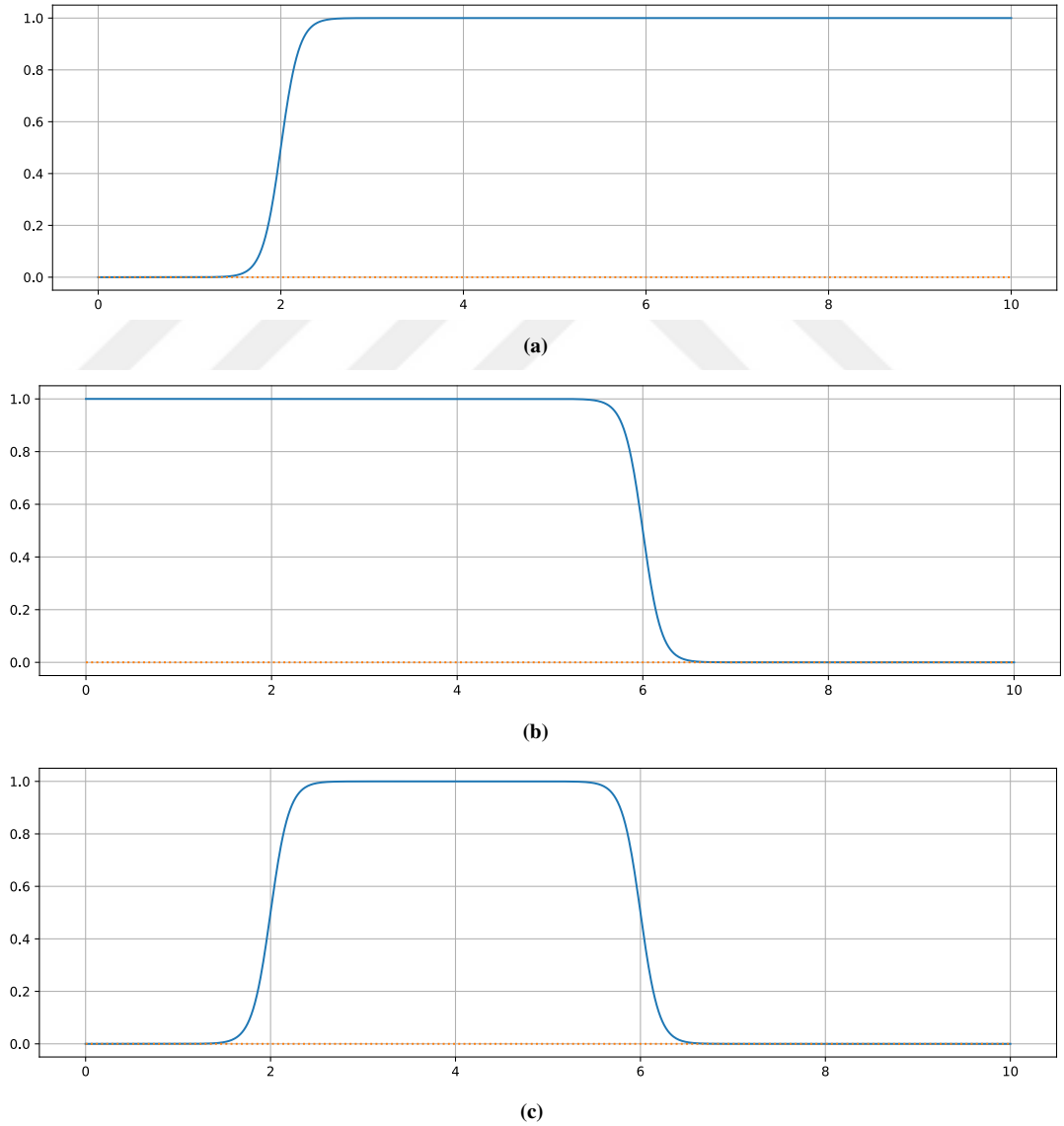


Figure 2.9 Three graphs show the sigmoid translated by 2 on the x-axis (a), the inverse sigmoid translated by 6 on the x-axis (b), and their multiplication result (c), which created a rectangular function

The auto-cropping part of the thesis uses translated and inverse sigmoids. Figure 2.9

shows how translated, translated and inverted, and their multiplication graphs look.

MLPs employ the backpropagation approach for their training phase. Figure 2.10 depicts the whole process of how an ANN trains. Text classification, speech recognition, medical diagnosis, and computer vision are just some of the tasks that have been tackled by ANNs, and backpropagation is the key to reaching these achievements.

Figure 2.10 An illustration of the various steps involved in the learning process of an ANN, including the initialization of weights, the forward propagation of inputs, the calculation of errors, and the adjustment of weights to minimize errors [12]

2.2.1 Backpropagation

While training an ANN, after a forward pass, the gradient needs to be calculated to update the weights. Backpropagation is employed to carry this out (e.g., gradient descent, which determines the gradient of the loss function, is employed by backpropagation) [92]. The error is propagated to previous layers and neurons, either

directly or indirectly linked to the output neuron (i.e., the neuron where the error is calculated), as depicted in Figure 2.11.

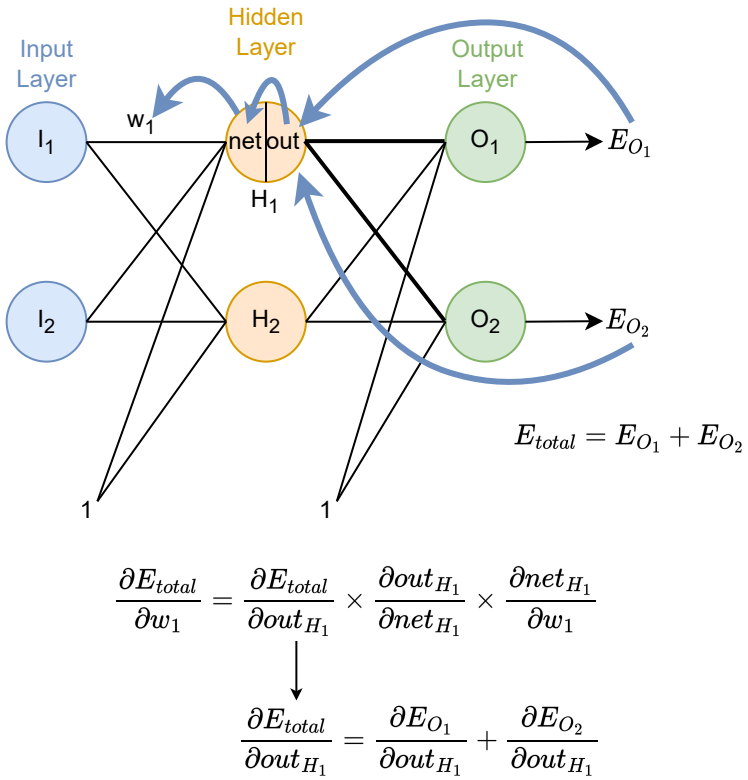


Figure 2.11 Demonstration of the process of backpropagation, which is for adjusting the weights in the network to reduce the error in the output by propagating the error back through the network during the training phase

Each neuron's net (i.e., incoming) and out (i.e., outgoing) values are determined while utilizing error analysis. The squared errors for the outputs are then determined. That can be stated as follows:

$$net_{H_i} = \sum w_j \times I_k + b \tag{2.1}$$

$$out_{H_i} = \frac{1}{1 + e^{-net_{H_i}}} \tag{2.2}$$

where I_k is the input to each w_j, w_j is the all weights linked to H_i

$$net_{O_i} = \sum w_j \times out_{H_k} + b \tag{2.3}$$

$$out_{O_i} = \frac{1}{1 + e^{-net_{O_i}}} \tag{2.4}$$

where H_k is the hidden neuron of each w_j, w_j is the all weights linked to O_i

$$E_{O_i} = \frac{1}{2} \times (target_i - out_{o_i})^2 \tag{2.5}$$

$$E_{total} = \sum E_{O_i} \tag{2.6}$$

2.2.1.1 Gradient Descent/Ascent Algorithm

The squared error cost function is minimized using gradient descent, where weights are revised after each iteration, and this process continues until the cost is as low as possible [88], as depicted in Figure 2.12.

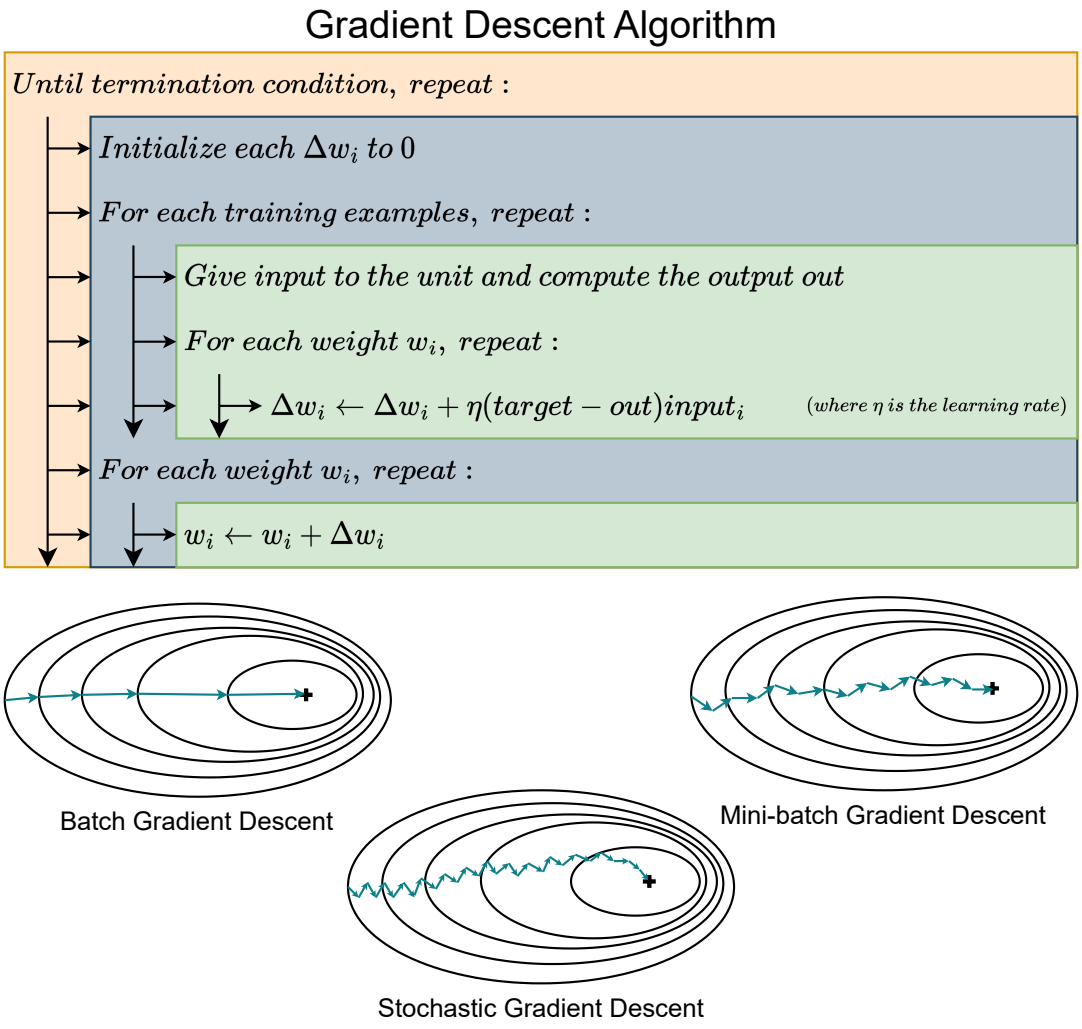


Figure 2.12 The algorithm of gradient descent optimization and an illustration of the comparison of different types

If the supplied learning rate is reasonably low, gradient descent is guaranteed to

converge to a conjecture with the smallest squared error. Gradient descent runs the danger of passing over the error surface's minimum if the learning rate is excessively high. Gradient descent could fail to reach the global optimum if the error surface contains many local optima. Additionally, reaching a local optimum might take so much time (i.e., potentially). To overcome these issues, variants of gradient descent known as stochastic gradient descent and mini-batch gradient descent were developed [88]. Figure 2.12 also depicts stochastic gradient descent and its comparison with batch gradient descent and mini-batch gradient descent (i.e., batch gradient descent tends to overlap the global optima as it updates after seeing the whole data, and stochastic gradient descent tends to get stuck at the local optima as it updates after seeing each sample).

Gradient ascent operates similarly to gradient descent, with a single distinction. It seeks to maximize a function, while gradient descent aims to minimize it. So, basically, it's the same optimization algorithm. As a result, the formula for gradient ascent is almost identical to the one for gradient descent (see Figure 2.12), only with an inverted sign (i.e., it will cause $(target - out)$ to transform into $(out - target)$).

2.3 Deep Learning

Deep learning is a subset of ML where the learning procedure occurs in deeper structures (i.e., as the term "deep" indicates). Deeper structures imply the presence of several hidden layers. While deep networks may contain tens or even hundreds of hidden layers, traditional NNs only have one or two [9, 10]. Large NNs can be considered to be performing a DL procedure as they are deep networks. DL can be employed for both unsupervised and supervised learning.

Nonlinear processing units are extensively used in DL structures. They are fundamental because the layers utilizing these units perform transformations and feature extraction operations. In the same way as NN, every layer's input is the output of the preceding layer, where each layer represents the abstraction differently. Each layer alters its input to produce an abstracter representation, where different levels of abstraction can be achieved by varying the size and quantity of layers.

In supervised learning, unlike ML, DL eliminates the requirement for manual feature extraction by converting the data into intermediate feature representations as it extracts features first-hand from the data itself.

ML approaches require the use of specialized methods such as feature extractors or human feature engineering, as well as extensive pre-processing procedures; in contrast, the stages of feature extraction and modeling are automated and require minimal or no pre-processing in DL. This fundamental difference is illustrated in Figure 2.13.

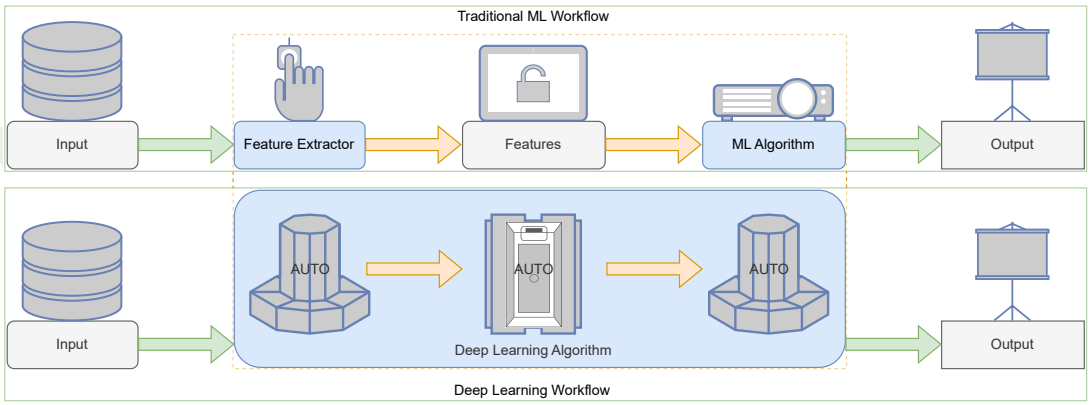


Figure 2.13 A visual comparison of the workflows of traditional ML and DL, highlighting the key difference

Another standout benefit of DL is its capability to continuously enhance its performance as the amount of training data grows. Large amounts of data and strong GPUs are now commonly accessible due to technological progress. This circumstance has increased DL's popularity. Selecting one of the ML techniques over the DL ones is more appropriate only when dealing with sparse datasets or weak computing environments.

The latest DL studies have been extremely successful, exceeding humans in tasks such as reading lips [93], playing games such as Go [94] or StarCraft II [95], object classification (i.e., where a dedicated human labeler made errors at a rate of 5.1% while Microsoft's system made errors at a rate of 3.5%), and diagnosing medical images [96].

In 1986, Rina Dechter introduced the phrase "Deep Learning" [97], then LeCun et al. developed a DNN that could read handwritten ZIP codes from mail using the backpropagation approach in 1989 [3]. However, SVMs, or Gaussian Mixture Model

(GMM)-HMMs, were more popular back in the day because of the high processing cost of ANNs. As time went on, advances in GPU technology became more significant, making DL considerably more popular than other methods. The "Big Bang" occurred in 2009, when Nvidia trained DNNs on Nvidia GPUs [98].

NNs form the basis of the majority of all DL techniques (i.e., the overview can be seen in Figure 2.14). Even though some other learners exist, the most recent and successful DL algorithms rely on NNs such as Deep Belief Networks (DBN), Recurrent Neural Networks (RNN), and CNNs. Because of this, DL models are frequently referred to as DNNs, and they have been used in a wide variety of fields, such as voice recognition, image recognition, text classification, and medical diagnosis.

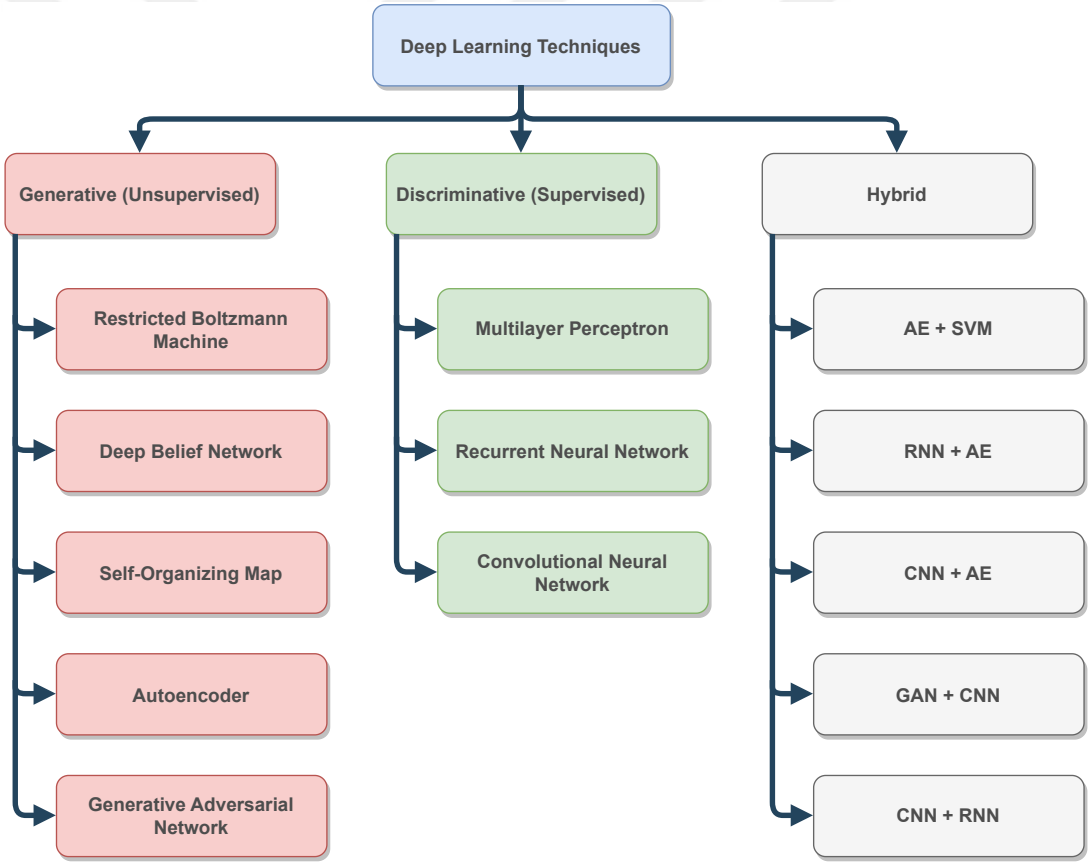


Figure 2.14 An illustration of various techniques used in DL

If there are several hidden layers between the input and output layers, that network is called a DNN [99], and the comparison is depicted in Figure 2.15. As DNNs are feedforward networks, data travels from the input layer to the output layer [12]. DNNs have a strong modeling capability since they can sort out linear or non-linear

connections, even if they are exceedingly complex. When modeling complex data, putting additional layers in hidden layers (i.e., unlocks getting the combination of features from previous layers) may reduce the number of units required in each hidden layer [100].

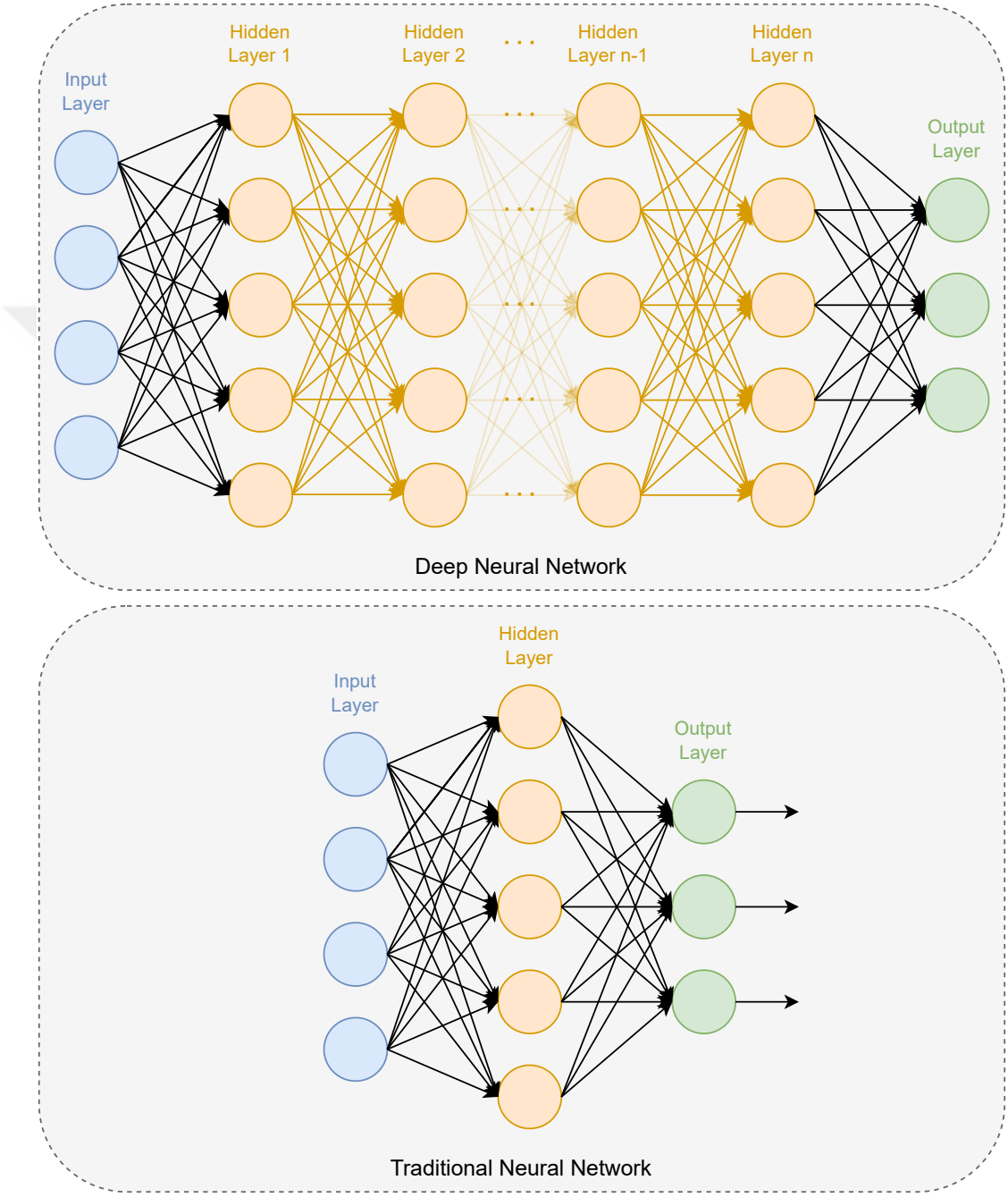


Figure 2.15 The differences between a DNN and a traditional NN by comparing their architectures. It depicts the number of layers and the complexity of the networks

In general, DNNs are very challenging to train for supervised learning. However, CNN is a notable exception for training deep networks [100].

2.3.1 Convolutional Neural Network

Neocognitron, a multi-layered and hierarchical ANN, was invented in 1980 [101] and was influenced by the Hubel & Wiesel paradigm [102]. It has been utilized for a variety of pattern recognition applications. It is regarded as the source of inspiration for CNNs.

Backpropagation was utilized by LeCun et al. to learn the convolutional kernel coefficients from photographs in 1989 [3]. The challenge was to recognize the handwritten ZIP code numbers that were depicted in the images. In contrast to manual coefficient design, the method learned coefficients from photos automatically and performed better.

A 7-layered convolutional neural network called LeNet-5 [103] was introduced in 1998 by LeCun et al. to recognize digits from 32×32 photos. For processing higher-resolution pictures than 32×32, the network needed to include additional and bigger (i.e., more layers with more nodes in each layer) layers. However, the resolution was limited to 32×32 due to the limited hardware capabilities available at the time, as increasing it would result in high processing power requirements.

CNNs attracted great attention after the computing industry acquired advanced hardware capabilities. Several studies have utilized and demonstrated how to train CNNs on GPUs and their appropriate approaches [6, 104–109]. Nowadays, CNN is one of the most prominent and effective DNN types, which are highly used in computer vision applications (i.e., clustering approaches, image recognition, or video recognition).

CNN is a type of DNN that is commonly used for image classification tasks. It is composed of input layers, output layers, and multiple hidden layers, and works by transmitting input data over the network. The CNN is designed to minimize the need for pre-processing of the input data, making it a very convenient choice for many applications. Additionally, CNN does not require manual feature engineering (i.e., a process of extracting relevant features from the input data), which again makes it a popular choice among researchers and practitioners. The CNN also has the success of

achieving state-of-the-art results on many image classification benchmarks and can be re-trained for new data or tasks.

When people view a picture of a cat, they can identify it based on its unique features, such as its claws, four legs, tail, and whiskers [110]. Similarly, a CNN can classify a picture of a cat by processing the low-level features, such as curves and edges, and then creating more abstract concepts using multiple convolutional layers [111]. Unlike some other machine learning algorithms, CNN does not require the explicit input of relevant features. Instead, it can automatically capture the necessary information from the input data to make a classification.

Image recognition models that utilize traditional MLP architectures often encounter difficulties when attempting to accurately classify higher-resolution images due to the "curse of dimensionality," a phenomenon in which the number of weights required by the model increases exponentially with the size of the input image. This occurs because MLPs have full connectivity between nodes, meaning that each node is connected to every other node in the subsequent layer. As a result, a fully connected layer of a 32×32 input image requires 1024 weights, while a fully connected layer of a 224×224 input image requires a much larger number of weights, 50,176. In contrast, convolutional layers in a CNN can operate with a much smaller number of weights, even when processing large input images. For example, a 7×7 filter that convolves on a 32×32 or 224×224 image will always require only 49 learnable parameters, regardless of the size of the input image. This efficiency makes CNNs a more practical choice for image classification tasks, particularly when dealing with high-resolution images.

CNNs are composed of layers with three-dimensional neurons, each of which is connected to a small region of the previous layer known as the receptive field, as illustrated in Figure 2.16. This structure allows CNNs to operate with fewer weights compared to traditional MLP architectures, as the connections between neurons are more localized and not fully connected.

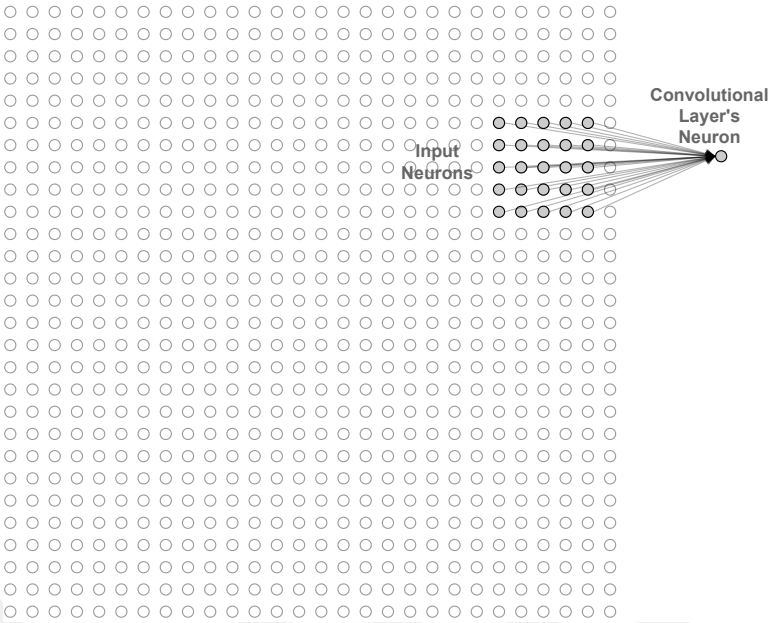


Figure 2.16 An illustration of the concept of a local receptive field (5×5 pixels), which is a small region of an image (28×28) that is connected to a neuron in the next layer of a DL model and is used to extract features from the image

CNNs differ from traditional ANNs and DNNs in the types of operations performed by their hidden layers. These layers typically include a combination of convolutional layers, pooling layers, a softmax layer, fully connected layers, and Rectified Linear Units (ReLUs). The convolutional layer is always the primary component of every CNN. On the other hand, the other layers are inserted between convolutional layers, except the fully connected layers, which are typically placed at the end of the network. These hidden layers serve to introduce non-linearity and maintain the dimensions of the input data. An illustration of a CNN that is used to classify the input image as one type of living species (i.e., animal species and human) is depicted in the Figure 2.17.

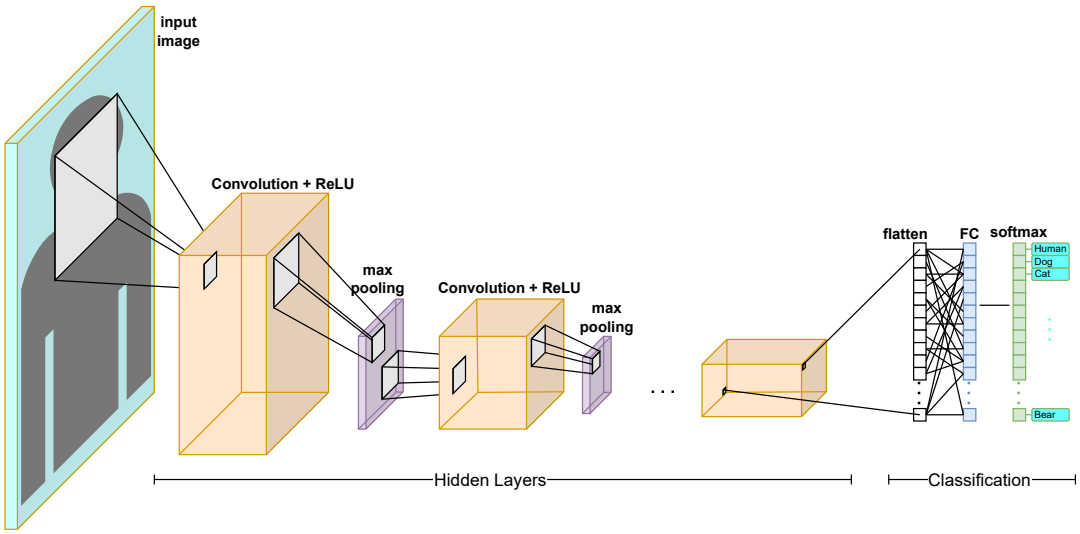


Figure 2.17 The architecture of a CNN, including the input image, convolutional layers, pooling layers, and fully connected layers

2.3.1.1 Convolutional Layer

A key component of a CNN is the convolutional layer. It is characterized by a set of learnable filters, which are the parameters that are adjusted during training. These filters are used to scan the receptive field, a small region of the previous layer, to search for matching patterns. The filters are represented as rectangular arrays of numbers that serve as feature identifiers, helping the CNN to identify and extract important features from the input data. Convolutional layers can be found multiple times in a CNN, including at the beginning as the first layer (i.e., the first layer after the input is always a convolutional layer) and play a crucial role in the network's ability to classify the input data.

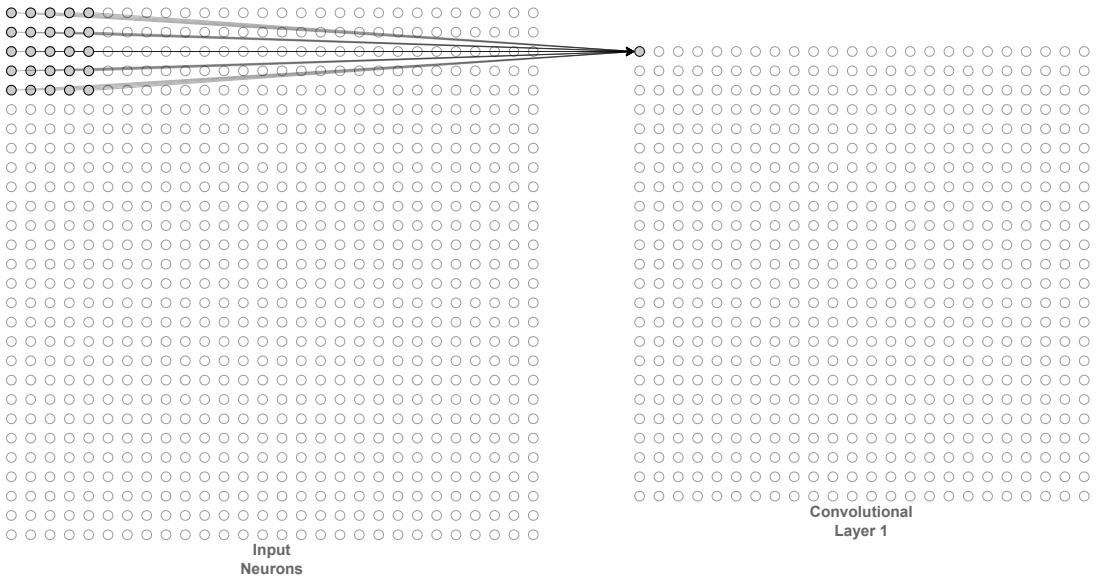


Figure 2.18 An illustration of the local receptive field (5×5 pixels) of an image (28×28 pixels). It shows how it is connected to the corresponding hidden neuron

In order to extract useful information from an input image, a CNN applies a small matrix of values, known as a filter, to the input. An example of a 5×5 filter can be seen in Figure 2.18 and Figure 2.19 sliding across the width and height of the input image (i.e., one pixel at a time). The process begins by moving horizontally across the width of the input. Once the end of the width is reached, the filter moves down one pixel vertically and starts again at the beginning of the next horizontal line. This continues until the filter has been applied to every location in the input. In order to process the input using this technique, a minimum of 24×24 hidden neurons are required (i.e., $(28-5)/1+1=24$). Each of these neurons is connected to a 5×5 region of the input and calculates the dot product between the entries in its corresponding filter and the input. This produces an activation map of the filter, which represents the output of that particular convolutional operation. The activation map is then used by the CNN to identify and extract features from the input image.

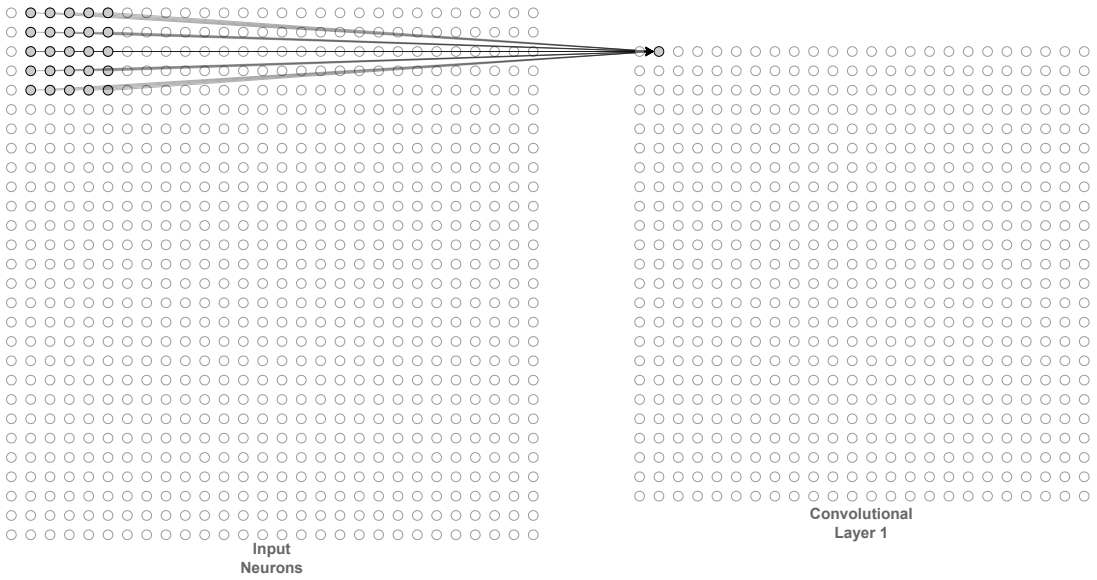


Figure 2.19 An illustration of the local receptive field (5×5 pixels) of an image (28×28 pixels). It shows how it is connected to the corresponding hidden neuron when slid by 1 pixel

With a stride length of 1 and a filter size of 5×5, the figures show the movement of the filter across the input image. The stride determines the distance at which the filter moves, and changing the stride length will result in a different number of hidden neurons being used. For example, if the stride length is set to 3 and the filter size is 7×7, there will be 8×8 hidden neurons for a 28×28 input image. However, in the current case with a stride length of 1 and a filter size of 5×5, the use of 24×24 hidden neurons is required for the same input image size. The number of hidden neurons required for a given input size and filter configuration can be calculated using a simple formula (see Equation **2.8**).

If an RGB image is used as the input, the input layer will have a size of 28×28×3, the filter size will be 7×7×3, and the first hidden layer will have a size of 1×8×8. In contrast, when the input is a single-channel image, the input layer size is 28×28, the filter size is 7×7, and the first hidden layer size is 8×8. It is worth noting that if there are two filters instead of just one, the volume of the first hidden layer will be 2×8×8 in the case of an RGB image. The number of hidden neurons needed can be determined based on the stride length, filter size, and input image size.

An additional parameter that can be used in a convolutional layer is padding, which increases the size of the input volume while the filters are being applied. When zero

padding is employed, the input volume is surrounded by a border of zeros. For example, if zero padding of size 2 is applied to a 28×28 input layer, the input size will be increased to 32×32, as depicted in Figure 2.20. If the stride length is selected as 1 when using 5×5 filters on the padded input, the output of this operation will be 28×28, which would be 24×24 if zero-padding were not utilized. It is evident that the actual input size is 28×28, while the size of the first convolutional layer remains unchanged at 28×28. Padding is often used to preserve the spatial dimensions of the input and control the spatial resolution of the output.

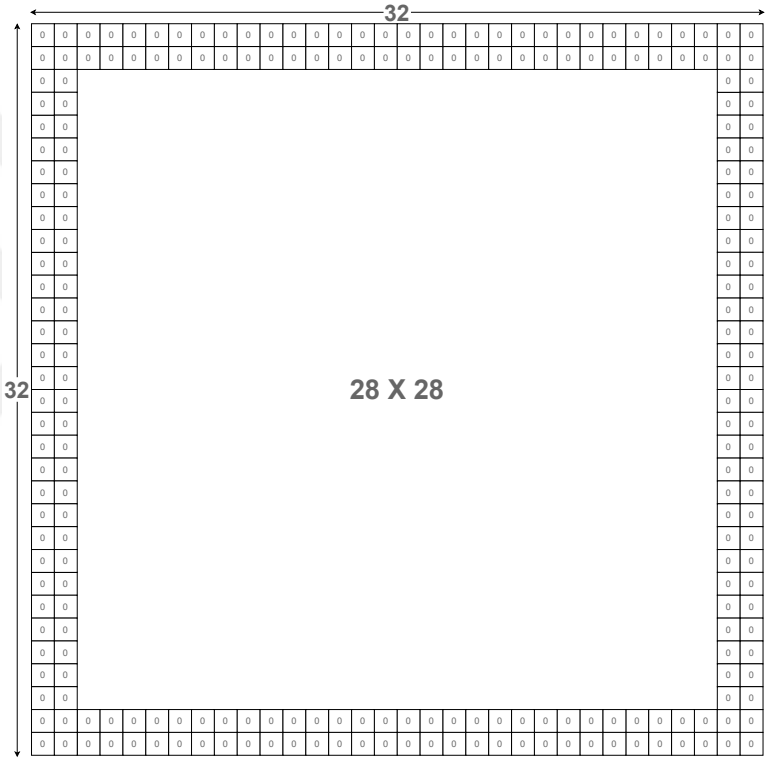


Figure 2.20 An illustration of the concept of zero padding, where additional rows and columns of zero values are added to the edges of an image (28×28 pixels) before it is processed by a CNN. In this case, 2 pixels of zero padding are added to the input neurons

The activation of filters in a convolutional layer occurs when they identify certain features in the input. Through this process, the network is able to learn the activation maps for each filter (i.e., effectively learn how to identify features). Generally, the beginning convolutional layers are responsible for identifying basic (i.e., low-level) features such as edges, curves, corners, and colors (i.e., in this study, it is grayscaled, so the value of gray is the low-level feature). The later ones tend to focus on more complex features (i.e., high-level combinations of features). This can be observed in

the illustrations of the first, second, fourteenth, and twentieth convolutional layers of a CNN (i.e., ResNet101) shown in Figure 2.21, Figure 2.22, Figure 2.23, and Figure 2.24, respectively. The first convolutional layer contains filters that represent only simple features such as colors and edges (see Figure 2.21). When the network progresses deeper, the convolutional layers start containing increasingly detailed filters that are able to identify features such as curves, corners, and combinations of edges (e.g., the second convolutional layer as depicted in Figure 2.22) of the object. In the final layers, these filters may even be able to recognize significant parts of the entire object (i.e., tooth-like shapes can be seen in Figure 2.23 and Figure 2.24). The visualization of a fully connected layer in Figure 2.30 consists of combinations of teeth.

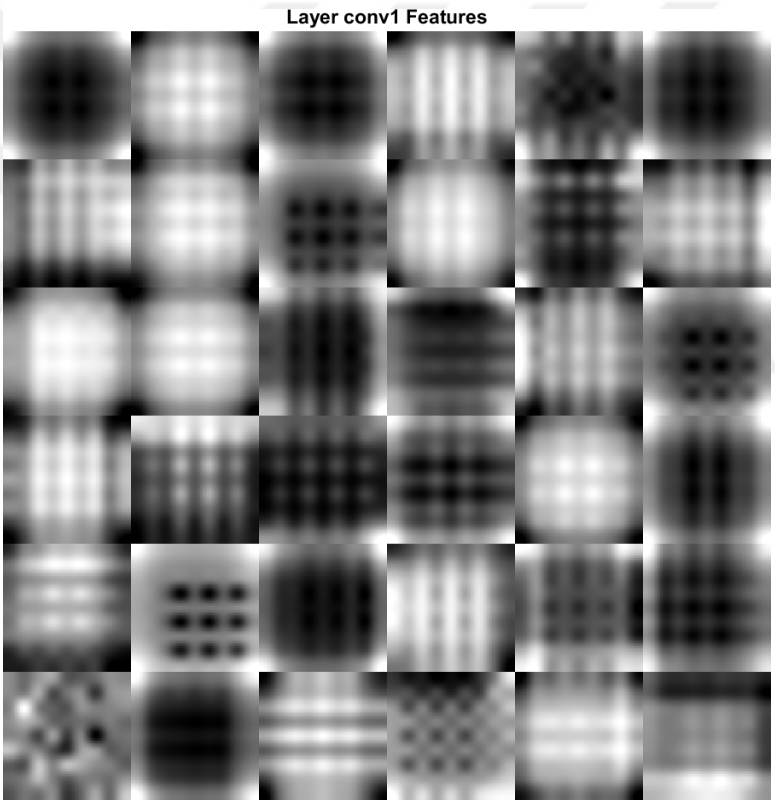


Figure 2.21 A visualization of the features detected by the first convolutional layer after the network has been trained by displaying the filters and the corresponding feature maps. It also shows how the first convolutional layer is composed of a set of filters; each filter is responsible for detecting a specific feature in the input image after the network has learned from training data

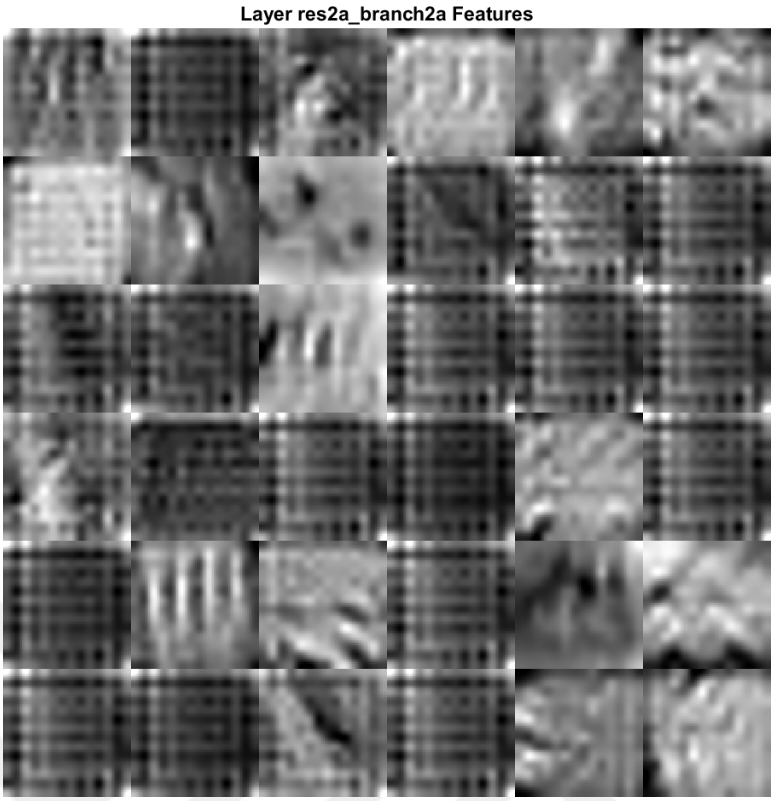


Figure 2.22 A visualization of the features detected by the second convolutional layer

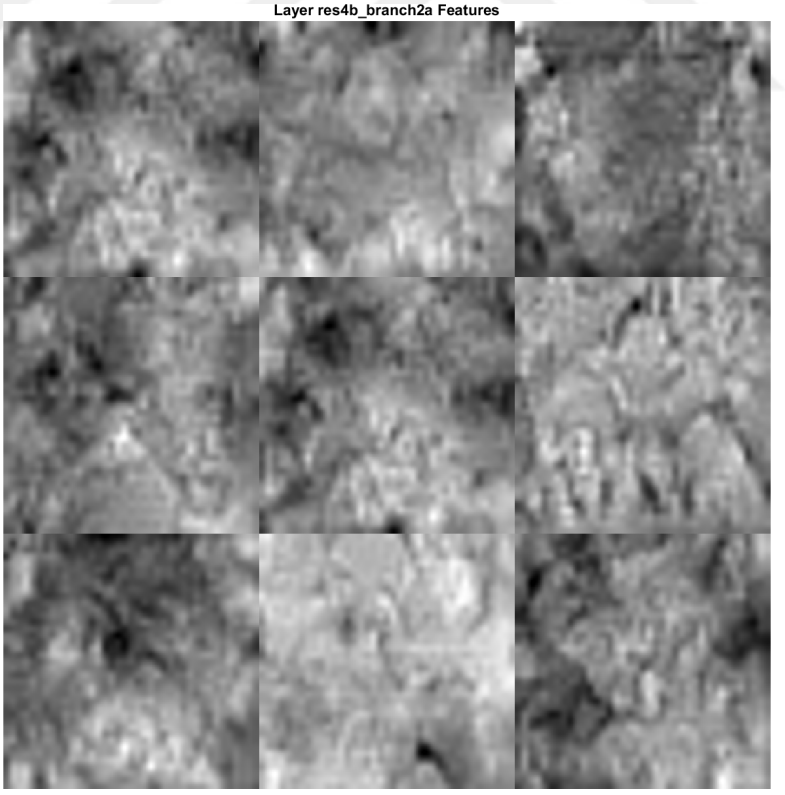


Figure 2.23 A visualization of the features detected by the fourteenth convolutional layer

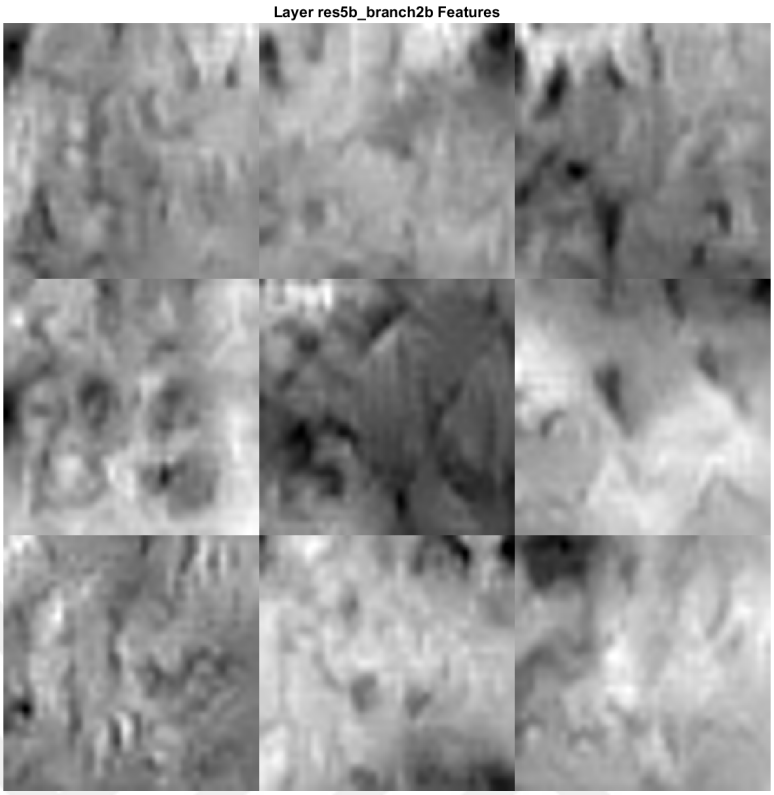


Figure 2.24 A visualization of the features detected by the twentieth convolutional layer

2.3.1.2 ReLU Layer

ReLU is a type of activation function that is widely used in deep learning models. It is common for a ReLU layer, which introduces non-linearity to the model, to follow each convolutional layer. It has been one of the most popular activation functions in neural networks since 2011 (i.e., release date). ReLU is defined as:

$$ReLU(x) = f(x) = max(0, x) \tag{2.7}$$

where x is the input of the activation function. If the formula is verbalized, then its result is the value provided as the input, or the value '0' when the input is less than '0'.

One of the main advantages of ReLU is its simplicity; its implementation is pretty straightforward and does not require additional hyperparameters. Additionally, ReLU has improved the training speed of deep neural networks compared to other activation functions (e.g., sigmoid or hyperbolic tangent). ReLU also helps to alleviate the vanishing gradient problem, which occurs when the gradients of the weights in the network become too small (i.e., training becomes slow).

Despite its advantages, ReLU has a serious problem that can cause a phenomenon known as "dead neurons" (i.e., the output of the activation function is always 0, resulting in the neuron not contributing to the model's predictions). The problem can be mitigated by using a variant of ReLU (leaky ReLU), which allows for a small negative slope when the input is negative.

Using a ReLU layer introduces non-linearity to the network, while convolutional layers primarily perform linear operations (i.e., multiplication and summation). While sigmoid and tanh are other options for introducing non-linearity, ReLU is preferred due to its computational efficiency (i.e., ability to train faster than other activation functions) [6]. Additionally, ReLU can increase non-linearity without significantly impacting the receptive fields or accuracy, and it can help to prevent the vanishing gradient problem (i.e., the slow training of earlier layers). The ReLU function converts all negative activations to zero, as shown in Figure 2.25.

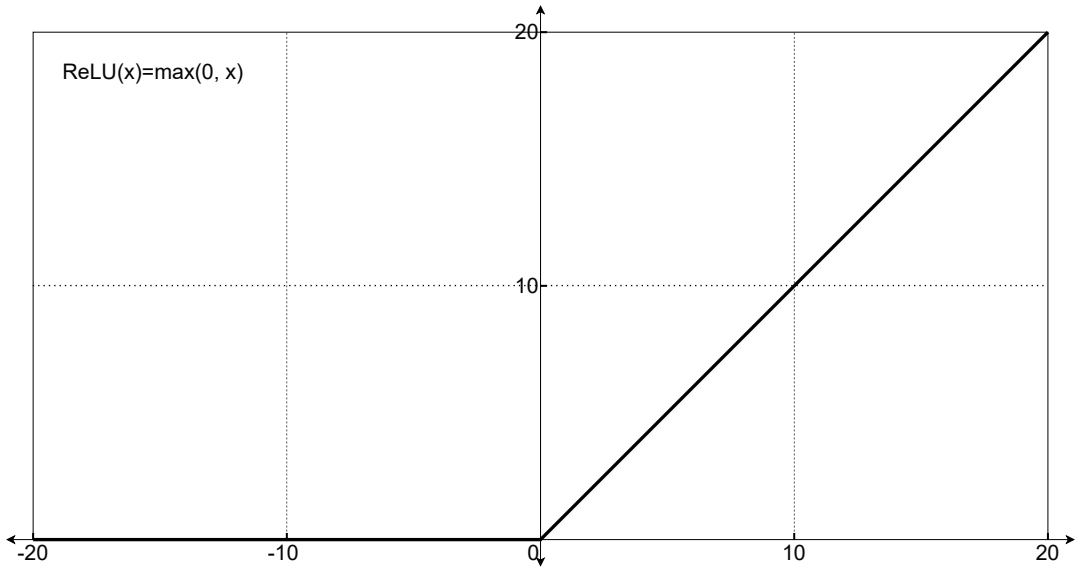


Figure 2.25 A graph that displays the ReLU activation function. It shows how the function returns the input value if it is positive and returns zero if it is negative

2.3.1.3 Pooling Layer

Pooling layer is an essential component of traditional CNN architectures. The majority of them are used right after the convolutional layers. Their duty is to reduce the spatial dimensions of the results generated by the convolutional layers by merging the outputs of multiple neurons into a single neuron that utilizes non-linear functions.

This simplification (i.e., downsampling) operation reduces the number of parameters and hence reduces computational overhead, as well as helping to prevent overfitting. Overfitting occurs when a model becomes too closely tuned (i.e., close to ideal or fully ideal) to the training data but fails on the test data (i.e., a generalization issue). Max-pooling is a widely-utilized non-linear function in pooling operations that partitions the input based on the filter size and stride length and selects the greatest value within each partition as the output. Min-pooling is the inverse version of max-pooling, where it selects the smallest value instead of the greatest one. Other pooling options include average-pooling (i.e., finding the average value in each partition) and L2-norm pooling (i.e., calculating the L2-norm of the values in each partition. The L2-norm is the distance of a vector from the origin).

In addition, pooling layers also help to maintain the spatial invariance of the network (i.e., they can recognize an object regardless of its position in the image) by using a pooling filter that slides over the input image. Sliding with the filter reduces the resolution while keeping the most important information. The spatial invariance is really useful in some applications, such as object detection, where an object can appear anywhere within the image. Furthermore, the pooling layer also acts as a regularizer, which helps to reduce overfitting by introducing a form of spatial sub-sampling that discards some of the information. In a CNN architecture, the pooling layer is generally implemented right after the combination of a convolutional layer and a ReLU layer.

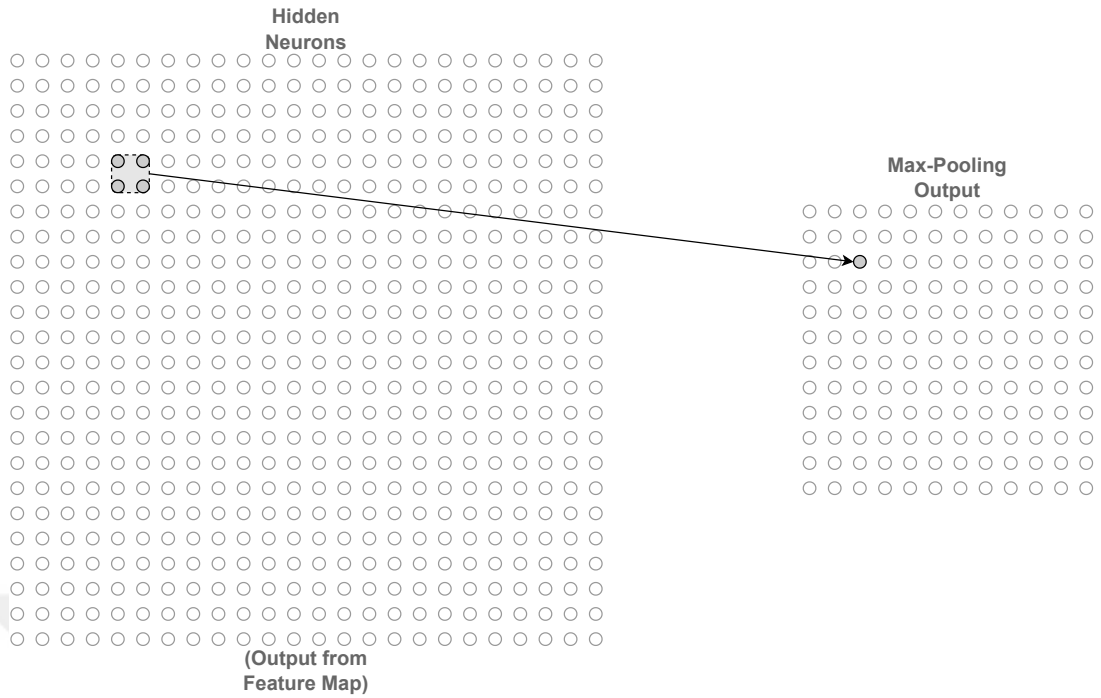


Figure 2.26 An illustration of the max-pooling, which is used to down-sample the spatial dimensions of a convolutional layer. The diagram shows the max-pooling operation applied to a 24×24 convolutional layer

Max-pooling, an example of a frequently used pooling layer, can be observed in Figure 2.26 (i.e., the filter size is 2×2 and the stride length is 2). The combination of filter size and stride in the Figure creates a 12×12 pooling layer, which can be calculated by the following equation:

$$pooling\ layer\ dimension = \frac{input\ size - filter\ size}{stride\ length} + 1 \qquad (\mathbf{2.8})$$

where it will lead to (24-2)/2+1=12.

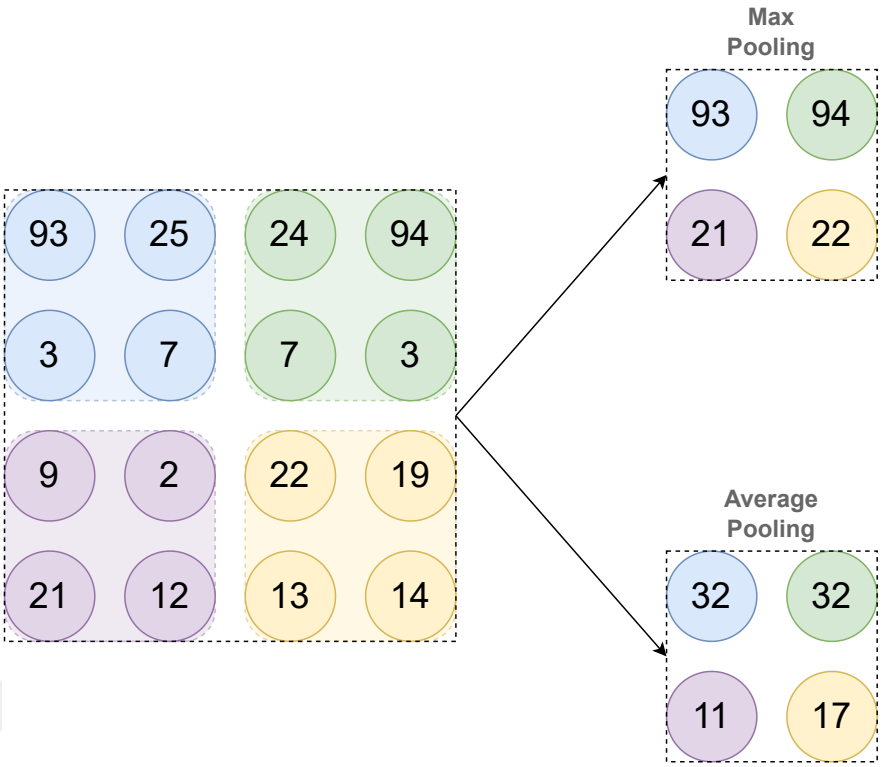


Figure 2.27 An illustration of the concepts of max-pooling and average-pooling. The diagram shows how max-pooling takes the maximum value from a small window of the convolutional layer, and average-pooling takes the average value from a small window of the convolutional layer

The required pooling operation (e.g., max-pooling, min-pooling, average-pooling, or L2-norm pooling) can be selected based on the specific requirements of the task and dataset (e.g., max-pooling is often used in image classification tasks, while average-pooling is used in tasks such as denoising). Similarly, RoI-pooling, a variant of max-pooling, is commonly used in object detection tasks. RoI pooling has a region proposal and a fixed output size (e.g., 2×2). It divides the region into sections of equal size and selects the maximum value from each section as the output [112].

Figure 2.27 depicts the use of both max-pooling and average-pooling operations. The left side of the illustration shows the result of a convolutional layer (with a size of 4×4), which is used as input for the pooling layer. The utilized filter size is 2×2, and the stride length is set at 2. As a result, the size of the pooling layer is 2×2 (i.e., (4-2)/2+1=2). When the 4 circles at the top left (i.e., inside the blue rectangle) enter the max-pooling operation, the highest number in that partition, 93, is selected as the resulting value of that filter. If they enter average-pooling, the average of those 4 numbers is calculated, which results in 32 (i.e., (93+25+3+7)/4=32). 75% of the information is discarded by

downsampling the height and width by 2 in both Figure 2.26 and Figure 2.27.

2.3.1.4 Dropout Layer

Dropout is a handy regularization technique used to reduce overfitting in neural networks. It is a simple but effective method that can be used in any type of NN. It can be applied at different levels of the network (e.g., after the fully connected layer, after convolutional layers, or even before the final output layer), where the effectiveness of the dropout layer varies according to the application level.

This method consists of randomly dropping out certain activations in a layer, with a probability commonly set at 0.5. This implies that, for this probability value, half of the hidden neurons are dropped out randomly. The inputs and outputs of the dropped neurons are also removed during the training process (i.e., activations are set to 0, and incoming and outgoing connections are suspended). Once the training is completed, these neurons are recovered with their weights. This technique is beneficial for preventing overfitting by making the model more robust by diversifying the neurons and reducing their co-adaptation. Additionally, it can also improve the model's generalization capabilities by adding a form of regularization. The dropout technique is illustrated in Figure 2.28 in the context of a standard NN.

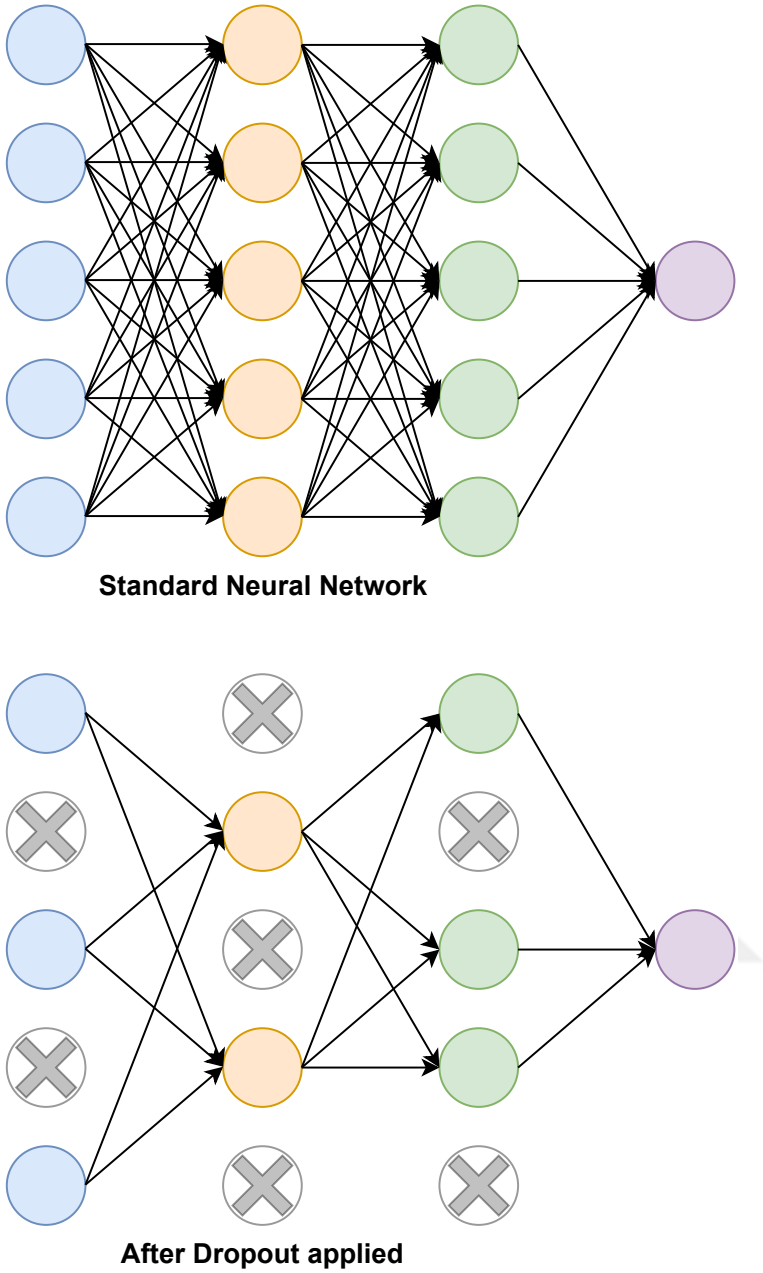


Figure 2.28 An illustration of the dropout which shows a NN without dropout and a NN after dropout

Dropout is a regularization technique that improves generalization by making the network more robust and less reliant on the training set. By randomly dropping out activations in a layer, the network is forced to rely on multiple neurons rather than just a few. This allows the network to perform well on the task even with a reduced number of neurons. During the training phase, the dropout layer is used to enhance the network's generalization ability by reducing the number of parameters in the network (i.e., decreasing computational cost and memory usage). However, during testing or validation, the dropout layer is not necessary as the network has already

learned to be robust during training. Dropout is commonly used in conjunction with other regularization techniques like weight decay and early stopping to further strengthen the network and prevent overfitting.

2.3.1.5 Fully Connected Layer

Fully connected layers, also known as dense layers, are typically placed at the end of a CNN to learn non-linear combinations of high-level feature activations that have been extracted through a series of convolutional and pooling layers (i.e., these layers take the output of the previous layer, which represents high-level features because they are at the end, and learn the most abstract and highest-level feature representations of the input by applying matrix multiplications and non-linearities). Furthermore, fully connected layers are also used to map high-level feature activations to the final output, making predictions or classifications.

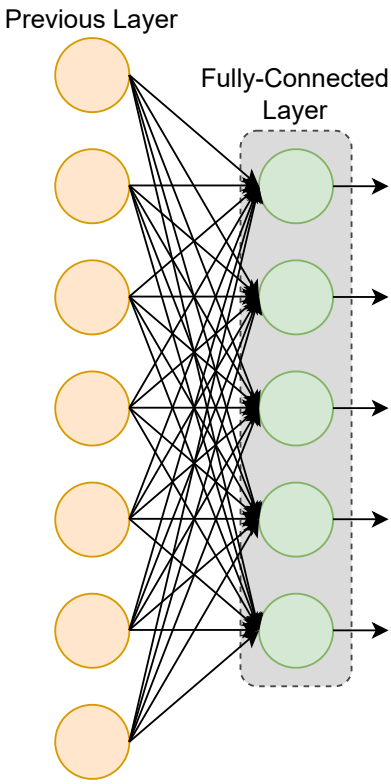


Figure 2.29 An illustration of the fully connected layer, which is composed of a set of neurons that are fully connected to the neurons of the previous layer

In the fully connected layer, every neuron is connected to all the activations of the previous layer (i.e., can be seen in Figure 2.29), which enables the network to combine

and mix important information from all preceding convolutional layers. This bears a resemblance to how traditional multi-layer perceptrons operate, where all of the neurons within a layer have connections to the preceding layer. In addition, fully connected layers are the ones that are going to make the final decision based on all the information, or predictions, collected from previous layers. As a result, the weights in these layers need to be trained very well to make accurate predictions.

In a fully connected layer, the input and output are both vectors, where each neuron computes a dot product of the input with its corresponding weight vector and applies an activation function. These layers can be stacked to form an MLP network. The chosen number of neurons in this layer defines the output dimension.

The visualization of a fully connected layer that has features of teeth (i.e., a combination) is depicted in Figure 2.30.

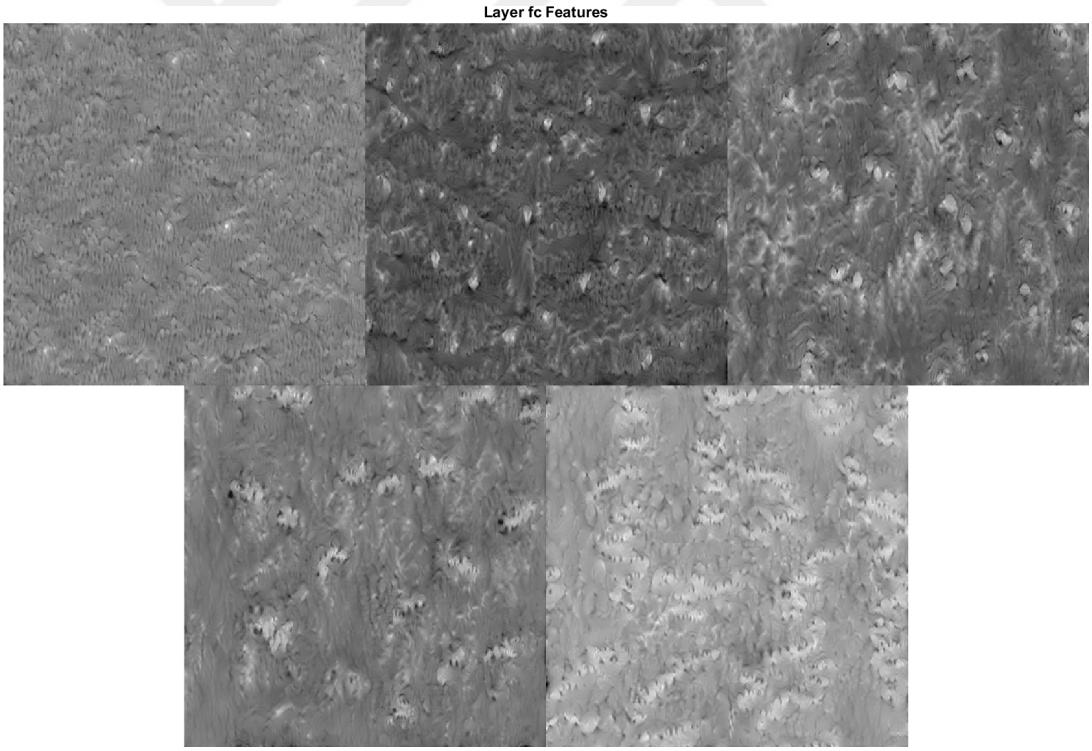


Figure 2.30 A visualization of the features detected by the fully connected layer

2.3.1.6 Softmax Layer

The softmax function is a generalization of the sigmoid function, which is used for binary classification. The Softmax layer's main duty is to perform multi-class

classification. The softmax function (i.e., a probability distribution function) inspired the name of this layer. The softmax layer takes the output of the previous layer as input, and through the softmax function, it produces probabilities of each class (i.e., indicates the output class that the input is most likely to belong to). The output of this layer can be represented as a probability distribution over the possible classes. It is typically placed at the end of a CNN as the last layer. It is often followed by a loss layer to calculate the difference between the predicted class probabilities and the true class labels to compute the final classification error. In Figure 2.31, how a softmax layer reacts at a forward pass is illustrated by showing the probabilities of three classes as its output.

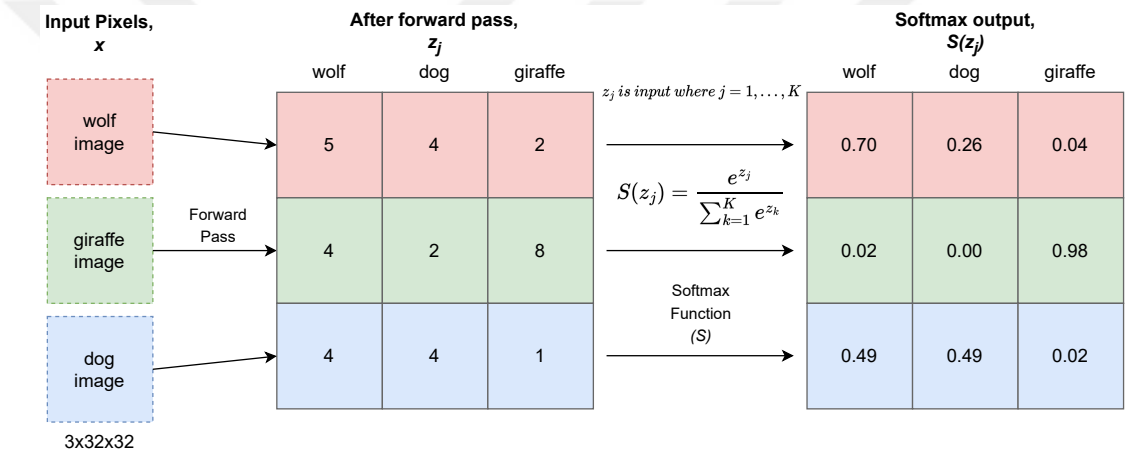


Figure 2.31 An illustration of the softmax function and example of the softmax output which is a probability distribution over the classes (wolf, giraffe, dog), with the class with the highest probability considered as the output of the network

The network's forward pass is completed once the forward pass through the softmax layer is complete. Then the network calculates an error by comparing the predicted values with the target values. The most widely used algorithm for updating the weights and biases in a CNN is gradient descent (see Section 2.2.1.1). It is an iterative optimization algorithm that uses the calculated error to adjust the network's parameters in a direction that reduces the error. This process is called backpropagation (see Section 2.2.1), and it allows the network to learn and improve its performance over time.

The softmax function takes multiple score values as input and compresses them into values ranging from 0 to 1, where the sum of all compressed values equals 1. The

output of the softmax layer can be interpreted as a probability distribution (i.e., the sum is 1) over the classes (i.e., the highest probability indicating the predicted class), which is ideal for producing probabilities of the possible outcomes. The softmax function can be implemented using matrix operations and applied to the entire batch of inputs at once, making it computationally efficient.

It is worth noting that in the context of deep learning, the softmax layer is not always the last layer; it can also precede other layers such as dropout, normalization, and so on. This choice depends on the complexity of the problem, the architecture of the network, and the desired outcome.

2.3.2 Residual Neural Networks

The vanishing gradient problem is a significant obstacle while training DNNs. It happens when the network's gradients get very small as they backpropagate through the network, which can block the network's ability to learn. This is a major issue since it prevents the network from learning well, especially when the network is very deep.

A residual neural network, also known as ResNet, is a type of NN that has been exclusively designed to address the vanishing gradient problem in DNNs. ResNets were introduced by He et al. [10] in 2015. The use of residual blocks helps to address this difficulty by allowing gradients to flow much more readily across the network. This is because the residual function is much easier to optimize than the actual function being learned (i.e., the mapping or transformation that the NN is trying to approximate).

ResNets contain residual blocks, which can be considered as the key part of the network. These blocks enable the network to learn the residual function rather than the actual function (i.e., being learned) by adding a shortcut connection to the block before passing through the rest of the block. The key benefit of this approach is allowing the network to learn much deeper architectures, as the gradients can flow much more easily through the network. The basic structure of a residual block is shown in Figure 2.32.

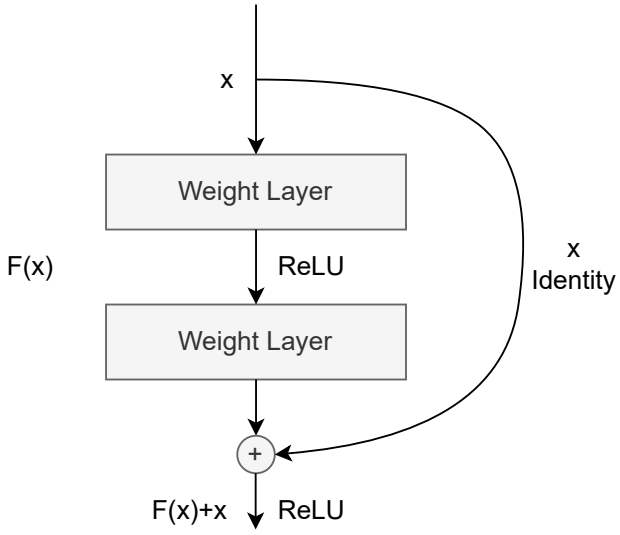


Figure 2.32 Basic structure of a residual block [113]

In the residual block, 'x' is the input of the block, '$F(x)$' is the residual function being learned, and "$F(x)+x$" is the output of the block. The shortcut link allows adding the input directly to the output of the block before processing the rest of the network.

The basic structures of convolutional residual block and identical residual block are shown in Figure 2.33.

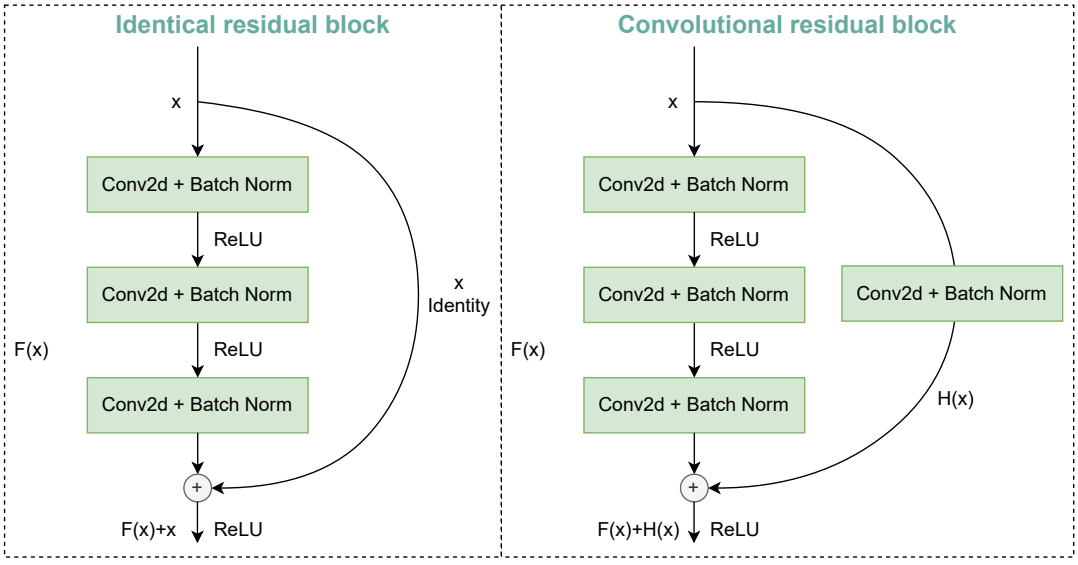


Figure 2.33 Structure of convolutional residual block and identical residual block

In the convolutional residual block, 'x' is the input, '$F(x)$' is the residual function (i.e., about to be learned), and "$F(x)+H(x)$" is the output. The shortcut connection contains a convolutional layer for matching the output dimensions of the shortcut path and the

main path.

ResNets also include batch normalization to help stabilize DNN training by normalizing network activations. Batch normalization has been commonly used in various cutting-edge NN designs. The main advantage of batch normalization is that it helps to avoid the internal covariate shift (i.e., the change in the distribution of network activation caused by changes in network parameters during training [114]) that greatly slows down the training process. The overall structure of a ResNet (i.e., ResNet-50) is shown in Figure 2.34.

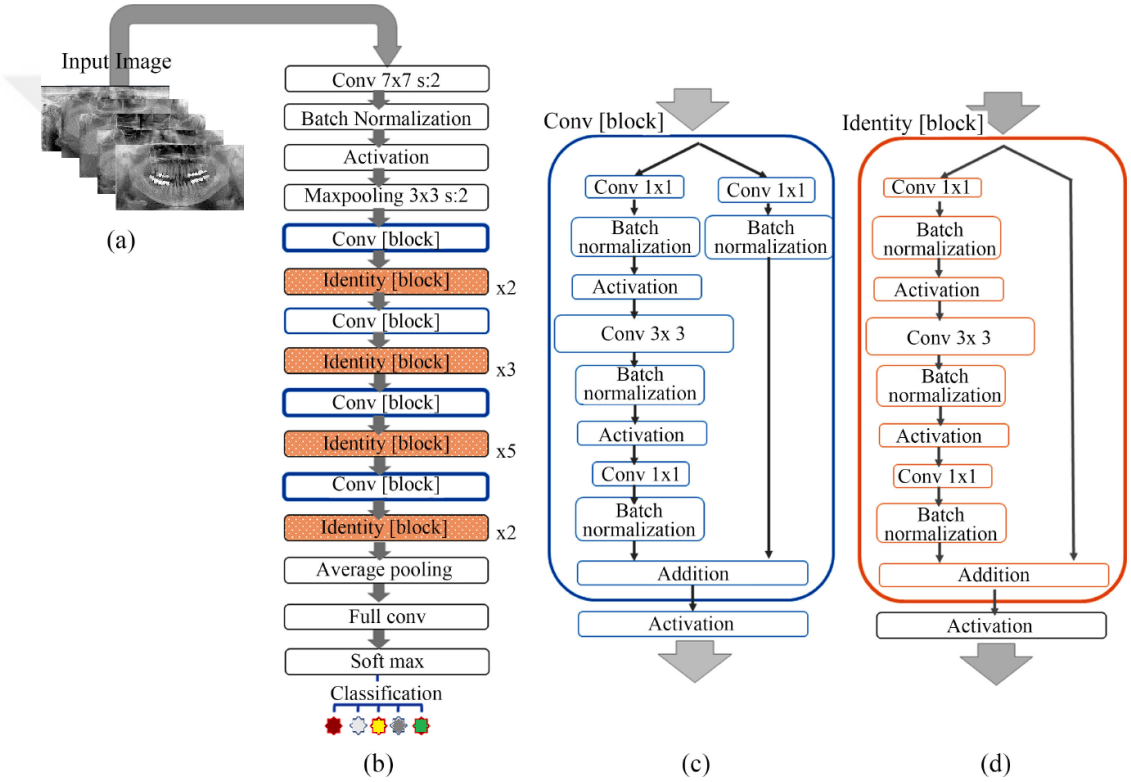


Figure 2.34 Outline of ResNet-50 architecture: (a) The input (grayscaled image). (b) The overall structure of the ResNet-50 architecture (the symbols ×2, ×3, and ×5 in the figure represent the number of blocks). (c) The composition of a convolutional block (the dimension of the input has changed at the shortcut). (d) The structure of an identity block (the output of the shortcut remains the same). [115]

In Figure 2.34, the input of the network goes through a series of residual blocks, which are connected by shortcut connections. The output of the final residual block then passes through a final fully connected layer, which produces the final output of the network.

ResNets can achieve very good performance on a wide range of tasks, including image

classification, object detection, and semantic segmentation, due to their ability to preserve learning capability in deep architectures (i.e., effectively capture the underlying structure of the data being processed). ResNets are known to be more memory and computationally efficient than traditional feed-forward networks. The main reason for efficiency is the use of residual blocks that allow the network to reuse activations and gradients from previous layers. Additionally, ResNets are known to be more robust against adversarial examples (i.e., crafted inputs designed to fool the network) as ResNets have multiple non-linear transformations. Also, ResNets have been used to improve the performance of other models like Generative Adversarial Networks (GANs) and RNNs.

In conclusion, ResNets are powerful and effective for addressing the vanishing gradient problem in DNNs. They have been widely adopted in a variety of applications and are particularly effective at learning DNN architectures. The use of residual blocks and batch normalization allows ResNets to effectively learn DNN architectures and achieve state-of-the-art performance on a wide range of tasks.

CHAPTER 3

IMPLEMENTATION

Deep learning architectures like CNNs require a significant amount of data to achieve reliable performance and a high-performance GPU to train in a reasonable amount of time, as more data means a longer training time. The deeper the network is, the more parameters it contains, and CNNs require further input data and greater computation resources. It can be challenging to provide effective results when CNN is trained on a tiny dataset, yet it is meant to be employed because of its benefits. To overcome this challenge, data augmentation can be used. Even when a large dataset is available, data augmentation can still be useful to increase the diversity of the data and improve performance.

3.1 Dataset

The availability of large and diverse datasets is crucial for achieving successful results in ML. Even with advanced algorithms, the most significant progress is often made through the expansion of the data available for training and testing.

Datasets are the biggest obstacle to the success of ML in domains with little or no data. This thesis focuses on utilizing a custom-built dataset, unique to the specific domain, to apply a successful DL method.

The dataset for this study was obtained retrospectively from the Ankara Yıldırım Beyazıt University (AYBU) Tepebaşı Oral and Dental Health Training Hospital. The study design and protocol were approved by the AYBU non-drug ethical committee, with the approval date and number as 19/04/2019-12. The study was conducted in compliance with the Helsinki Declaration of 1964, as updated in 2013.

The dataset, made up of panoramic X-ray radiographs, comprises over 20,000 unique samples. Despite the possibility of some samples originating from the same person, taken at different times, or even on the same day, every radiograph is unique. This is due to the fact that each individual may have different restorations at different times,

and the conditions under which the X-ray was taken can vary, such as the angle of the head and body. Therefore, there is no duplication in the dataset, and every sample can be utilized by the chosen DL method.

The panoramic radiographic image dataset was retrospectively acquired from the Picture Archiving Communication System (PACS) (Infinitt PACS, developed by Infinitt Co., Seoul, Korea) of the dental hospital. The dataset was compiled over a period of three years (i.e., between January 2016 and April 2019) and subsequently classified and labeled using Electronic Medical Records (EMR). It is worth noting that the dataset comprises radiographs obtained from two distinct devices, specifically the Planmeca ProMax X-ray machine (Planmeca, Helsinki, Finland) utilized in the hospital. By using the same model of devices, the dataset is uniform compared to datasets employing different models of devices but also a bit more generalizable compared to datasets compiled using a single device.

In the initial stage of the process, all relevant images from the institution's PACS were retrieved, excluding those belonging to individuals under the age of 18. Subsequently, images that displayed distortion, scattering, or resolution issues were removed. The collection of X-ray images was then subjected to a thorough examination by three dental experts, who labeled each image into 33 different categories (i.e., initially). Following the validation process, only panoramic radiographs with fixed dentures that included crowns and bridges were selected for inclusion in the dataset. After the collection of X-ray images reached an adequate number of samples, the images were labeled. The dataset creation process is illustrated in Figure 3.1 (i.e., showing all steps). In summary, the aim was to obtain the maximum number of usable radiographs.

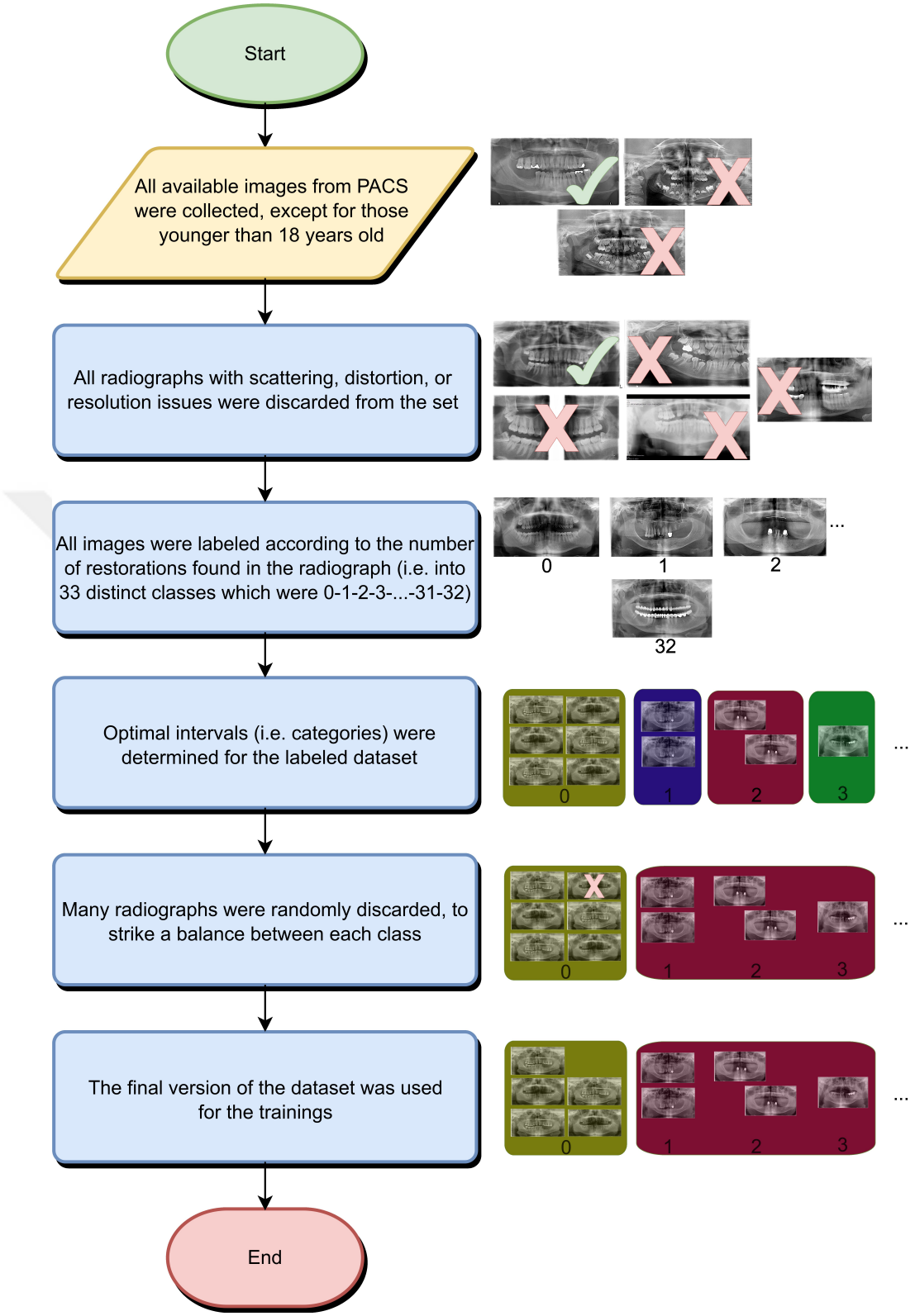


Figure 3.1 A visual representation of the various stages involved in creating a collection, including the steps of selecting, gathering, and cataloging the items. The illustration highlights the process of how the collection is compiled and organized, as well as the method of arranging and labeling the items within the collection [1]

The final annotation for each image was arrived at through a consensus-based process

among experts, and a comprehensive re-validation was conducted. Each image was classified based on the presence of fixed dentures. The dataset only includes panoramic radiographs with full crowns and bridges because they completely cover the abutment, making them radiopaque restorations, which are visible on radiographs regardless of the material used [25]. As a wide range of materials can be used for crowns and bridges, this is a distinctive feature for the purpose. Conversely, other restoration types such as inlays, onlays, and partial restorations were not included in the dataset because they exhibit similar levels of opacity [74, 116] and are indistinctive on radiographs.

From the initial collection of panoramic radiographs, a total of 20,973 samples were selected and classified into five distinct categories. The sample distribution was balanced by randomly excluding a portion of the initial collection (i.e., after the exclusion, every class had almost the same number of samples). The five categories are '0', '1-3', '4-6', '7-11', and '12-32', each representing a different quantitative level of dental restorations in a person's mouth (i.e., images with 4 or 5 or 6 restorations belong to the '4-6' class). The categories are defined as 'no restoration', 'low-level restoration', 'mid-level restoration', 'high-level restoration', and 'very high-level restoration'.

Table 3.1 Labeled categories and their respective characteristics [1]

Interval (# of Restorations)	# of Samples (%)	Category Name
0	4248 (20.25%)	No restoration
1 - 3	4231 (20.17%)	Low-level restoration
4 - 6	4227 (20.16%)	Mid-level restoration
7 - 11	4253 (20.28%)	High-level restoration
12 - 32	4014 (19.14%)	Very High-level restoration

As listed in Table 3.1, the dataset is composed of 4248 (20.25%) images of 'no restoration', 4231 (20.17%) images of 'low-level restoration', 4227 (20.16%) images of 'mid-level restoration', 4253 (20.28%) images of 'high-level restoration', and 4014 (19.14%) images of 'very high-level-restoration'.

At the pre-processing stage, all images were resized to a uniform resolution due to the networks only allow a definite input size (i.e., because of fully connected layers), so different resolutions do not pose a significant issue. However, it is important to

note that the resolution of the images varies, with 44% of the total images having a resolution of 2836×1536, 37% having a resolution of 2860×1536 and the remaining 19% having resolutions that are similar but not grouped into a single resolution. The auto-crop approach proposed in the thesis introduces scale-invariance property, so the varying resolutions are not a big deal again as it finds the RoI and resizes the image (i.e., bi-linear interpolation) into a fixed-size image without corrupting the aspect ratio.

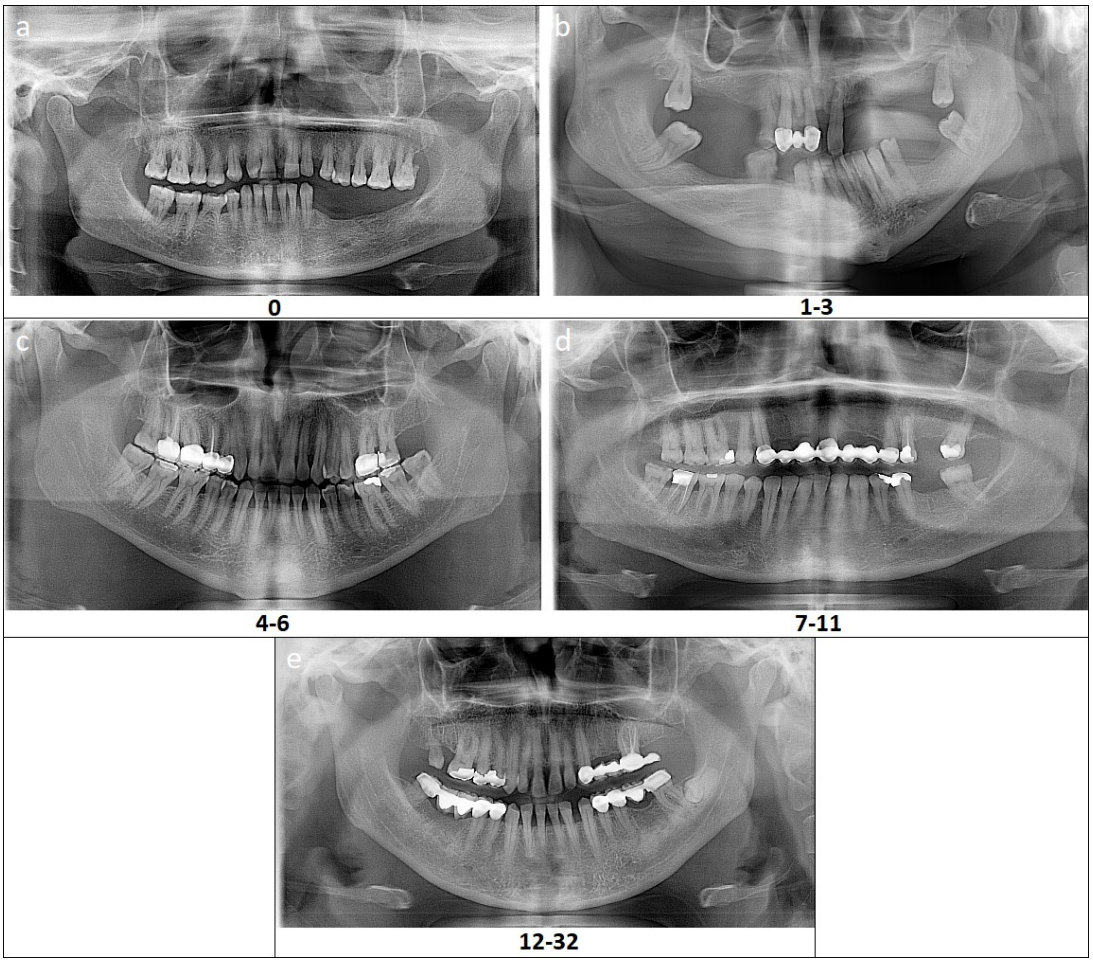


Figure 3.2 The images present a representative sample of each class, with '0' indicating 'no restoration' as demonstrated in image (a), '1-3' representing 'low-level-restoration' as illustrated in image (b), '4-6' denoting 'mid-level-restoration' as depicted in image (c), '7-11' exemplifying 'high-level-restoration' as portrayed in image (d), and '12-32' signifying 'very high-level-restoration' as shown in image (e) [1]

A representative example of each category is presented in Figure 3.2. The dataset is composed of 8-bit grayscale images, with a small proportion (i.e., 17%) stored in 16, 24, or 32-bit format (i.e., still display grayscale), which were transformed into single-channel during the pre-processing stage to achieve consistency.

3.1.1 Pre-Processing

The pre-processing stage includes converting all images to single-channel grayscale and resizing them to 224×224 pixels for all networks other than AlexNet and inceptionResNetv2 (i.e., 227×227 for AlexNet, 299×299 for inceptionResNetv2) for non-auto-cropped experiments. Experiments with the auto-cropping algorithm did not require extra resizing as it produced all inputs at a single resolution of 200×370 (i.e., the width is 370).

Image augmentation was also performed to achieve more reliable and accurate results as well as a more diverse dataset. An augmented training set was fed into the network with random parameters for each epoch (i.e., each time it is augmented randomly according to the parameters). The image augmentation included random translation between -700 to 700 pixels on the x-axis and -400 to 400 pixels on the y-axis (i.e., these parameters were the limits, because at most around 25% of the image can be lost in order to not lose features of the image as RoI falls to the center of the image as illustrated in Figure 3.3) for non-auto-cropped experiments. Additionally, random reflection on the x-axis, random rotation within an angle range of -15 to 15 degrees, random scaling between a factor of 0.8 to 1.3, and random shearing within an angle range of 0 to 15 degrees were employed for the augmentation for non-auto-cropped experiments.

Due to the uniqueness of each person, there is no universal RoI position available that fits all cases. As a result, earlier settings may excessively corrupt (i.e., after augmenting) certain samples. Translating and rotating without the information of the RoI midpoint leads to issues like the arrival of no-data (black) regions when augmented at limits, significantly reducing the dataset's effectiveness (i.e., since quite large areas become no-data regions).

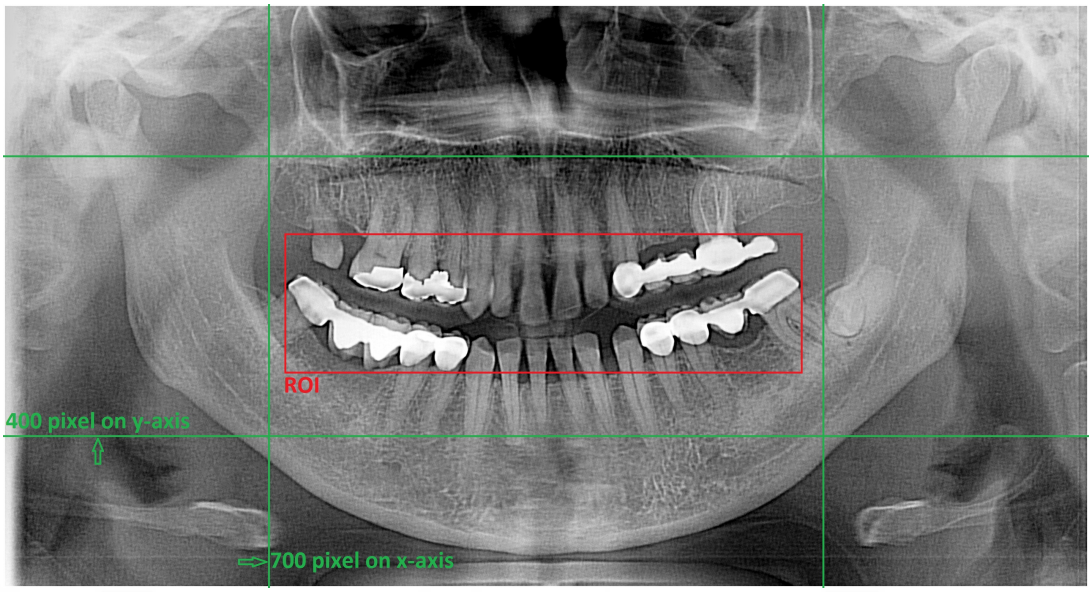


Figure 3.3 The image translation process, which centers the RoI within the image, was applied to each image, as exemplified by the fifth sample from Figure 3.2e [1]

Since the generated dataset is quite different from the original, a different data augmentation setting was used for the auto-cropped dataset. Augmentation of the auto-cropped dataset (i.e., the resolution reduced to 370×200) included random translation from −26 to 26 pixels on the x-axis, −14 to 14 pixels on the y-axis, random reflection on the x-axis, random rotation in the −7 to 7 degrees angle range, random scaling with a factor from 0.97 to 1.03, and random shearing over an angle range of 0 to 2 degrees.

Reflecting on the y-axis was not applied as it would lose the similarity information of the upper and lower jaws and surrounding structures such as the frontal and maxillary sinuses, nasal cavity, temporomandibular joint, and other nearby head and neck anatomy in each panoramic radiograph. Losing these details may reduce the accuracy of the training.

3.2 Auto-Cropping

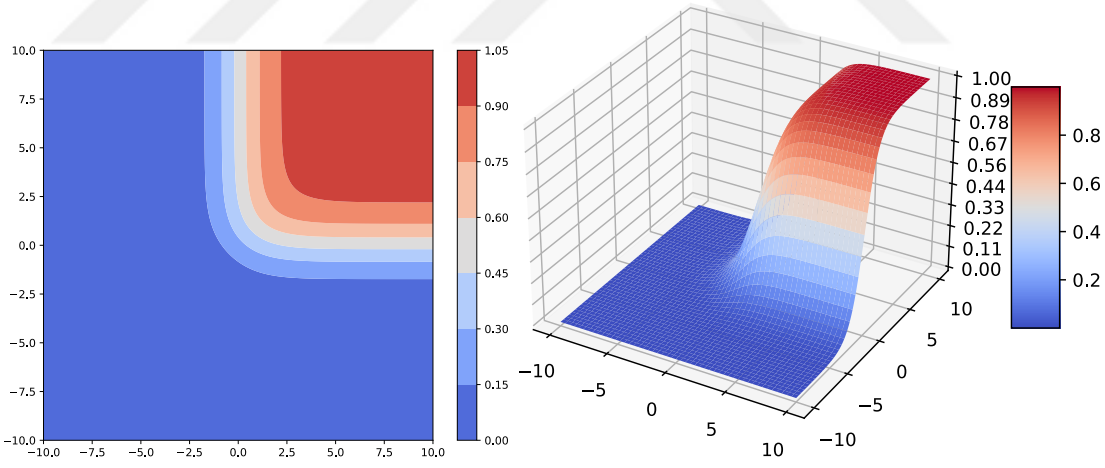
This section describes the auto-cropping optimization using the gradient ascent approach to maximize the cost function, which includes the dot product between a discrete signal (an image) and a continuous function (a rectangular function) to calculate the gradient and update the parameters in each iteration. It also mentions the

pre-processing step of taking the difference between max pooling and min pooling to smooth and downscale the input image.

3.2.1 Auto-Cropping Algorithm Overview

The RoI for the intended task is the person's mouth area, and its position is unique for each person. In addition, other structures in the panoramic radiograph should be removed for more successful training results. The most useful solution is to crop the relevant region from the images. An auto-cropping algorithm to select and crop the RoI from images was developed to fulfill this requirement.

Performing a dot product between a discrete image and a continuous function to calculate the gradient is essentially utilizing a numerical approximation of the derivative known as the inner product method or convolution. To make this happen, a continuous function was defined that represents the derivative to approximate. The function was discretized to match the sampling rate of the image, then the dot product of the discrete image and the discretized function were taken.



(a) Contour plot of the sigmoid

(b) 3D plot of the sigmoid

Figure 3.4 Plots of 2D sigmoid function

The introduced auto-cropping algorithm aims to iteratively optimize the parameters defining the RoI position in the image. The 2D sigmoid function (see Equation **3.1** and Figure 3.4) was used in the rectangular function due to its ability to emphasize salient

regions in the image while suppressing less important areas.

$$\text{2D Sigmoid Function: } \sigma(x,y,\gamma) = \frac{1}{1+e^{-x\cdot\gamma}} \cdot \frac{1}{1+e^{-y\cdot\gamma}} \quad \textbf{(3.1)}$$

where the default value of the sharpness (γ) is 1

The algorithm utilizes the gradient ascent optimization on the cost function to approximate the partial derivatives. Before that, a pre-processing step, which includes taking the difference between max pooling and min pooling, is applied to the image to blur and downscale it.

3.2.2 Pre-processing Step

3.2.2.1 Max-Min Pooling

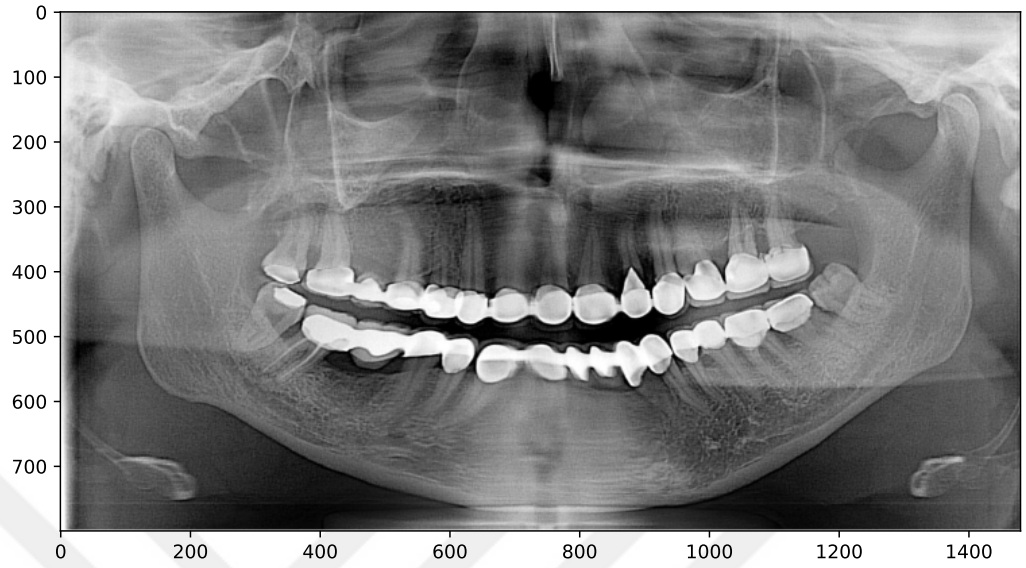
A differentiable pre-processing step was performed before the auto-crop optimization algorithm started searching the RoI on each input image. This step includes the calculation of the difference between the max-pooling and min-pooling operations (see Equation **3.2**). It is inherently differentiable because the max-pooling and min-pooling operations are differentiable (i.e., min-pooling is just the inverse version of max-pooling).

$$\begin{aligned} max\_min\_dif(x) &= max\_pool(x) - min\_pool(x) \\ &\rightarrow min\_pool(x) = -max\_pool(-x) \end{aligned} \quad \textbf{(3.2)}$$

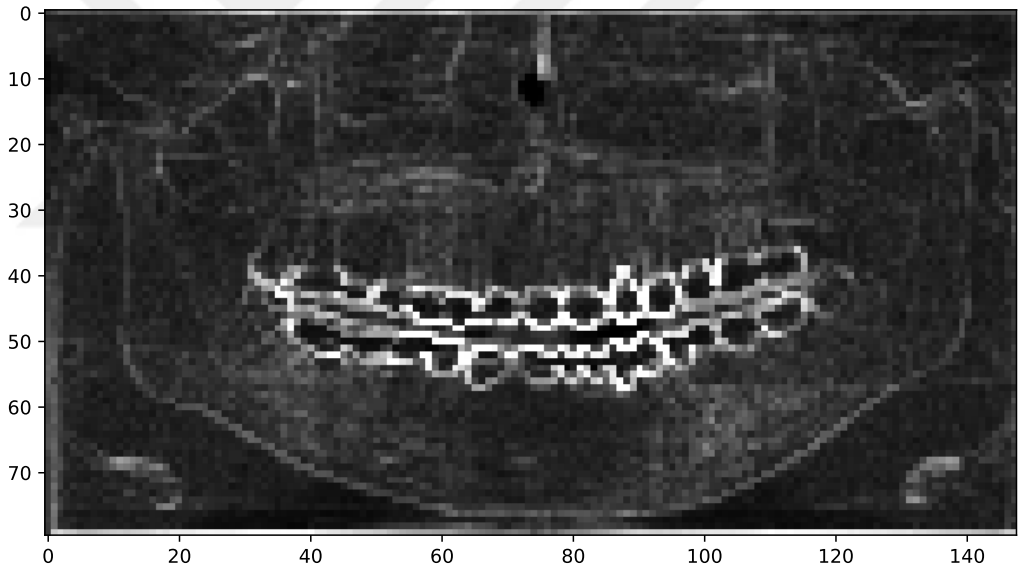
Pooling operations were used instead of just filtering (i.e., $max - min$ filtering) to reduce the resolution of the image. Max-pooling downsamples the image by selecting the maximum value within each pooling region (see Figure 2.26), while min-pooling selects the minimum value. Taking the difference of these pooled images helps to make the edges more detectable, blurs the image, and also reduces the size of the input image, making subsequent operations more efficient. The resulting image better reveals (i.e., clarifies) the significant differences in regional variability (i.e., clarity of the region) [117] for all teeth and serves as the input for the auto-cropping algorithm.

Both the pooling window size and the stride were selected as 10, which leads to downscaling the resolution to 10%. After finding the RoI parameters, every parameter

needs to be scaled by 10 to find the exact positions on the original image. Figure 3.5 shows how an input image can be transformed with a $max - min$ pooling operation.



(a) The original input with a resolution of 1480×800



(b) The resulting image with a downscaled resolution of 148×80

Figure 3.5 Before and after views of the image altered with $max - min$ pooling

3.2.2.2 Sobel Edge Detection

Similar results to the implemented $max - min$ pooling can also be achieved with an edge detection algorithm such as Sobel because $max - min$ pooling reveals edges and blurs regions with low regional variability (i.e., has similar effects to edge detection).

A Sobel edge detection algorithm was used instead of pooling operations to see an alternative result. It was desired to examine the difference between the pooling

operations, which are widely known in the DL community, and the Sobel operator, which is widely known in the image processing community.

Sobel edge detection is one of the most popular image processing techniques used to detect edges in digital images. This technique is a simple and effective way to highlight sudden changes in intensity or color in an image, often corresponding to object boundaries or edges.

Sobel edge detection works by convolving the image with a pair of 3x3 convolution kernels (also known as masks or filters). These two kernels are used to calculate the gradient of the image in the horizontal and vertical directions. One kernel is the other rotated 90°, as can be seen in Figure 3.6.

-1	0	+1
-2	0	+2
-1	0	+1

G_x

+1	+2	+1
0	0	0
-1	-2	-1

G_y

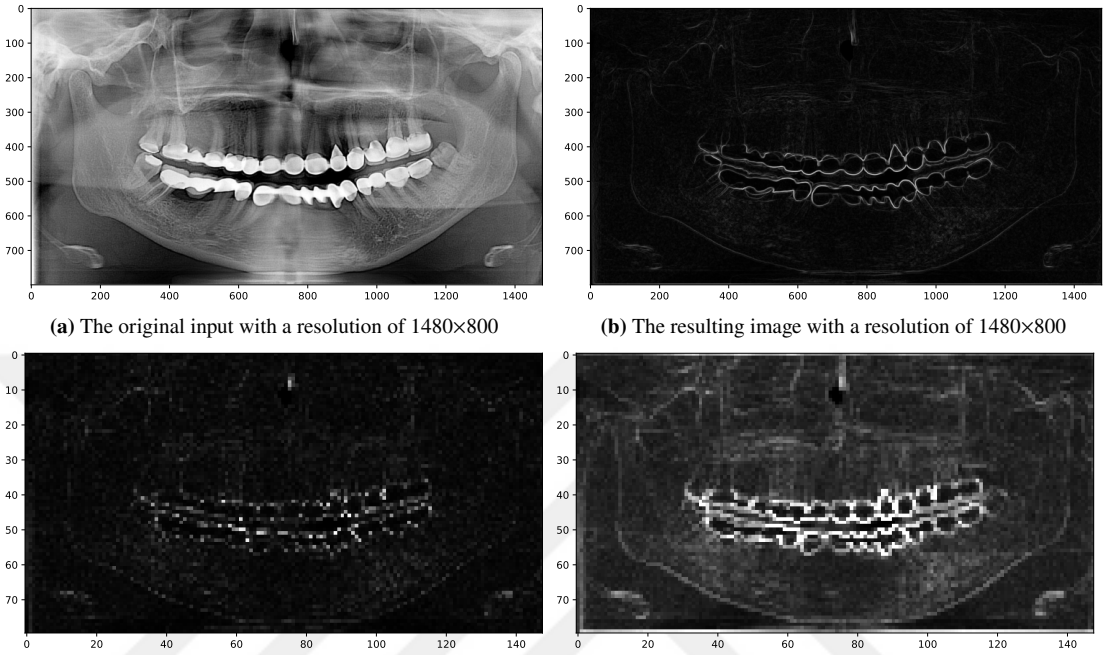
Figure 3.6 Sobel operators in the *x* and *y* directions

The gradient of the image in both horizontal and vertical directions can be calculated by applying the kernels individually. In order to compute the gradient magnitude at each point (i.e., the combination of both directions), the following was applied:

$$|G| = \sqrt{G_x^2 + G_y^2} \tag{3.3}$$

Before applying Sobel operators, Gaussian blurring was applied to eliminate or minimize unnecessary details that could lead to unwanted edges, while also preserving the overall structure.

After applying Sobel edge detection, the resulting images have the same resolution as the original images. They were resized using bi-linear interpolation to achieve the same resolution results as $max - min$ pooling. An example resulting image and its comparison with $max - min$ pooling can be seen in Figure 3.7.



(a) The original input with a resolution of 1480×800

(b) The resulting image with a resolution of 1480×800

(c) Sobel result downscaled to 148×80

(d) $max - min$ pooling result

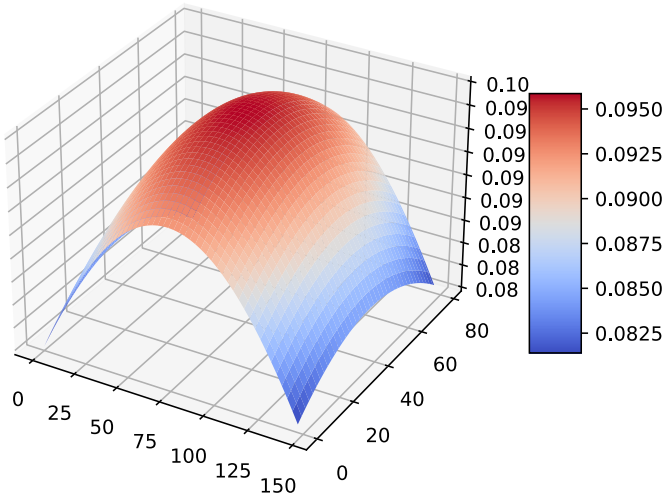
Figure 3.7 Before and after views of the image altered with Sobel edge detection and its comparison with $max - min$ pooling

3.2.3 Rectangular Function

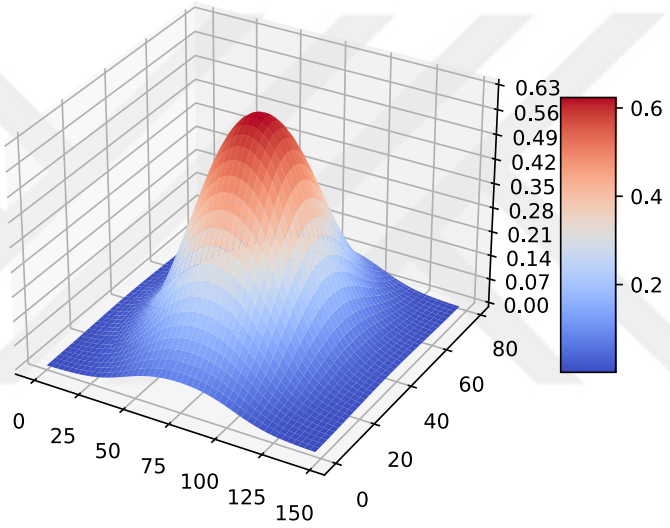
The rectangular function was defined according to the needs of the task. As the task is about finding the RoI for cropping, the rectangular function needs to search a rectangular area on the image. For this task, a rectangular function was employed, which can be produced using sigmoids as stated below:

$$\begin{aligned} &\text{Rectangular Function: } R\Big(x, y, x_c, y_c, r_{total}, \alpha, \gamma\Big) = A \cdot B \\ &A = \sigma\Big(x - (x_c - r_{total}), y - (y_c - \alpha \cdot r_{total}), \gamma\Big) \\ &B = \sigma\Big(-x + (x_c + r_{total}), -y + (y_c + \alpha \cdot r_{total}), \gamma\Big) \\ &\text{where } \alpha \text{ is the aspect ratio } (height/width) \text{ and the default value of } \gamma \text{ is 1} \end{aligned} \tag{3.4}$$

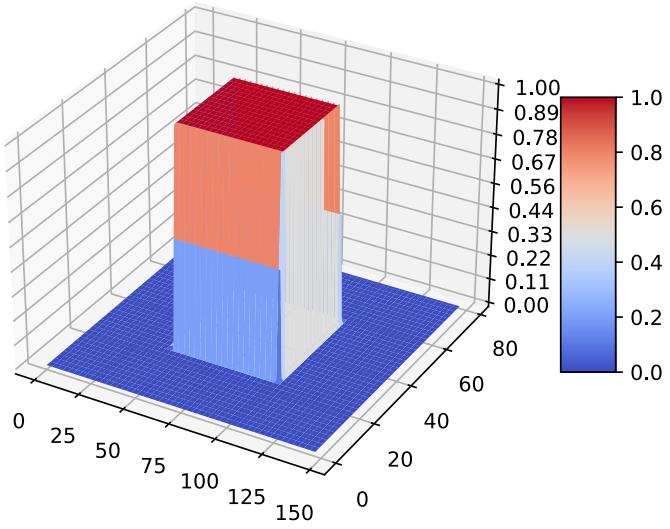
It is a combination of two sigmoid functions in 2D that creates a rectangular function (see Figure 2.9 for a 1D rectangular function created from 1D sigmoids). The parameters 'x_c' and 'y_c' define the central point of the RoI, and 'r_{total}' is the parameter for distance



(a) For the sharpness (γ) = 0.01



(b) For the sharpness (γ) = 0.1



(c) For the sharpness (γ) = 10

Figure 3.8 3D plots of the rectangular function discretized for a 148×80 image at its center $(74, 40)$ with $r = 30$ for different sharpness (γ) values

from that central point (i.e., $(\alpha \cdot r_{total})$ for y-axis). The parameter 'γ' controls the smoothness (or sharpness) of the rectangular function; as can be seen in Figure 3.8, a smoother rectangular function was obtained when 'γ' was reduced.

While optimizing with gradient ascent, a full-size rectangular function (i.e., the same resolution as the input image) is used. A function derived from the rectangular function is used in the cost function and will be explained in the next section.

3.2.4 Cost Function

The used cost function is not simply the inner product of an input image with the rectangular function; instead, an enhanced cost function derived from the rectangular function was used as follows:

$$f(x_c, y_c, r, r_{min}, s, x, y, \alpha, \gamma) = \frac{f_{inside}}{f_{outside} + 1} \tag{3.5}$$

$$f_{inside} = \frac{\sum_{i=0}^{n-1} s_i z_i}{\sum_{i=0}^{n-1} z_i} \quad \text{and} \quad f_{outside} = \frac{\sum_{i=0}^{n-1} s_i (1 - z_i)}{\sum_{i=0}^{n-1} 1 - z_i}$$

where 'z_i' is the rectangular function $\rightarrow z = R(x, y, x_c, y_c, r_{total}, \alpha, \gamma)$

'r_{total}' is "$r + r_{min}$" and 's' is the input image (3.6)

The numerator part of 'f_{inside}' is simply the dot product of the image and the rectangular function. The denominator part of 'f_{inside}' is the discretized rectangular function, which is a penalty to prevent (or slow down) 'r' from becoming infinitely big. As the name suggests, 'f_{inside}' ensures maximizing the inner part of the rectangular function (i.e., desired) while optimizing with the gradient ascent algorithm. Again, as the name implies, '$f_{outside}$' ensures minimizing the outer part of the rectangular function (i.e., unwanted) while optimizing with the gradient ascent algorithm; this also helps to maximize the 'f_{inside}'. The '1' was added to the denominator part of the cost function because when '$f_{outside}$' is between '0' and '1' and it goes to '0', the cost function goes to infinity. It can get the maximum value of 'f_{inside}' when 1 has been added to the denominator.

3.2.5 Parameter Update With Gradient Ascent

At each iteration, the algorithm computes the gradient of the cost function with respect to the RoI parameters (i.e., 'x_c', 'y_c', and 'r'). The gradient can be obtained thanks to the dot product between the image and the rectangular function. This inner product measures the correlation between the image and the rectangular function, highlighting regions of high similarity. The gradient ascent algorithm then updates the RoI parameters by taking a step proportional to the gradient, aiming to increase the cost function value and improve the RoI selection. This maximizing operation can be stated as:

$$\underset{x_c, y_c, r}{\operatorname{argmax}} f(x_c, y_c, r, r_{min}, s, x, y, \alpha, \gamma) \tag{3.7}$$

which means gradient ascent maximizes the cost w.r.t. x_c, y_c and r

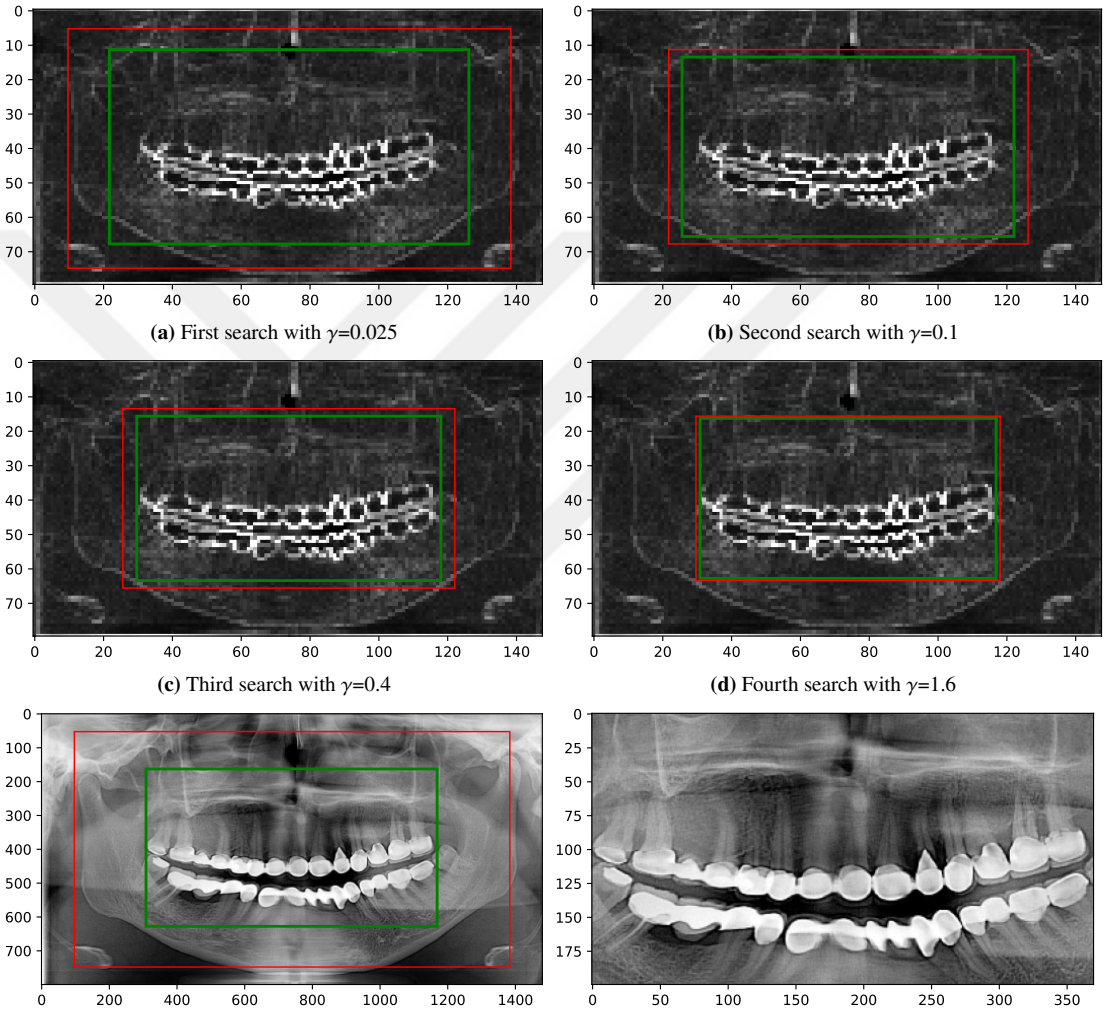
The parameters 'x' and 'y' are the vectors used for digitizing coordinates (i.e., sample points for discretization) of the cost and the rectangular functions to match the sampling rate of the input image.

3.2.6 Convergence, Stopping Criteria and Precautions

The auto-cropping algorithm iteratively updated the ROI parameters until a convergence criterion was met. Convergence was determined based on various factors, such as reaching a maximum number of iterations, hitting the vanishing gradient problem (i.e., explained in Section 2.3.1.2 and Section 2.3.2), or reaching a stable RoI position. To prevent some problems while converging, like an exploding gradient problem (i.e., the inverse of the vanishing gradient problem where the gradients get too large) or the update making RoI out of bounds (i.e., out of the image), some control mechanisms such as gradient clipping, shifting the center, and using absolute values were used.

The convergence criteria were 1e-4 for the vanishingly small gradients. For the exploding gradient problem, gradient clipping (i.e., rescales the gradients to keep them under control where it ensures the gradients have a norm at most $|\ the\_value\ |$) was applied to the exponent of the exponential function in the sigmoid function so that the single-precision floating-point format range limits (i.e., float32) were not reached

(i.e., for the sharper rectangular function, gradients were hitting the limits, so when the exponent was greater than | 50 |, it was clipped). The 'r' parameter was implemented to always take a positive value, and a 'r_{min}' parameter was added, which ensures that 'r_{total}' cannot be smaller than 'r_{min}'. If the parameter update caused the bounding box of the RoI to fall outside of the image, the 3 updated parameters were updated again to shift the center of the RoI, and thus the bounding box fell inside of the image again.



(a) First search with γ=0.025

(b) Second search with γ=0.1

(c) Third search with γ=0.4

(d) Fourth search with γ=1.6

(e) The initial and the last (optimized) bounding boxes

(f) Cropped and resized (370×200) result

Figure 3.9 Representation of the initial (i.e., red) and optimized (i.e., green) bounding boxes in the input image

3.2.7 Implementation Details

The auto-cropping algorithm was implemented using PyTorch [118]. The image processing operations, including max pooling, min pooling, and dot product, were efficiently performed using the Torch library. PyTorch was also selected to make the smart crop end-to-end due to its auto-grad capabilities. The concept of using

differentiable functions within the framework of PyTorch is quite straightforward due to its ability to optimize parameters directly through gradient-based optimization. Cropping can only be optimized by auto-grad if it is in a differentiable form. The steps taken to make the problem differentiable have been described in the previous sections.

The parameters of the rectangular function, such as the center ('x_c' and 'y_c') and the offset ('r', which affects 'r_{total}'), were fine-tuned in the later iterations to achieve optimal results for the specific task of RoI selection and cropping (i.e., 4 searches in total were held for each image). This is done by increasing the sharpness ('γ') and decreasing the step size (i.e., learning rate) after some rough searching (i.e., 'γ' initialized from 0.025 and multiplied by 4, step size initialized from 96 and divided by 2 after each search). Two RoI rectangles were illustrated for visual inspection in Figure 3.9 to see what the initial and optimized cropping regions look like in an input image. A rough search moved the bounding box a lot with lower sharpness and larger step size (Figure 3.9a). In addition, minor changes were observed in the fine-tuning stages (see Figure 3.9b, Figure 3.9c, and Figure 3.9d).

Initialization of the rectangular function's parameters (i.e., 'x_c', 'y_c', and 'r_{total}') was not done randomly. 'x_c' and 'y_c' were always initialized from the center of the image, and the 'r_{total}' offset was always initialized so that the bounding box of the RoI was near the edges of the image (i.e., 'r_{min}' was initialized to be $2/7 \times w$ and 'r' was arranged to make r_{total} to be $10/21 \times w$). These three parameters (i.e., 'x_c', 'y_c', and 'r') of the rectangular function were optimized by gradient ascent, as mentioned in Section 3.2.5.

3.3 Training Part

The breakthrough achievement of AlexNet [6] in the ILSVRC [11] with 63.3% accuracy sparked a flurry of efforts to enhance the performance of deep CNNs through modifications in parameters and the addition of layers to deepen the network [7–9]. However, the excessive deepening of the network made the training more burdensome and resulted in a decline in performance, as illustrated in Figure 3.10. The advent of deep ResNets [10] provided a solution to this dilemma by enabling the training of seriously deeper networks with ease. Therefore, ResNets led very deep

CNNs to a substantial improvement in performance (i.e., 77.15% and 78.25% accuracy with ResNet-50 and ResNet-101 on ImageNet, respectively).

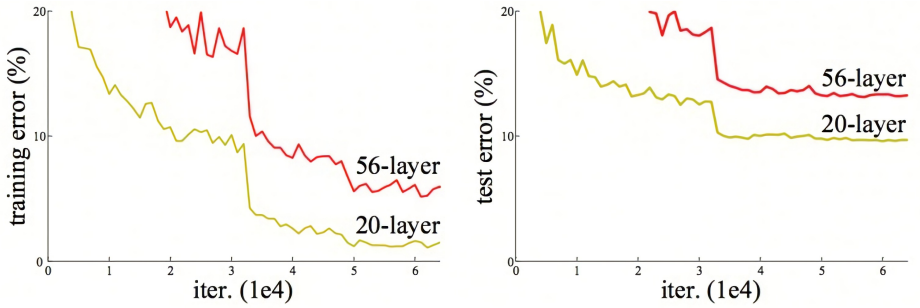


Figure 3.10 The comparative analysis of the performance of 20 and 56 layered plain DNNs on the CIFAR-10 dataset [119]. The y-axis shows the error rates and the x-axis represents the number of iterations [10]

The outstanding success of ResNet sparked a renewed interest among researchers in the DL community to refine the architecture and incorporate other techniques in pre or post-training phases, resulting in the emergence of even more successful variants such as Inception ResNet V2 [120] with 80.1% accuracy and BiT-L [121] with 87.54% accuracy on ImageNet.

Variants of ResNet such as ResNet-18, ResNet-50, ResNet-101, and Inception ResNet V2 were experimented with in this thesis. Additionally, the utilization of AlexNet, considered as the foundation of deep learning, and VGG-16, considered as its successor, were tested. The selection of these networks was based on their historical significance in performance leaps on the ImageNet dataset in ILSVRC [11] and also on their continued success and widespread use (i.e., ResNets are still one of the most successful networks in society). Furthermore, the excess of data available in the dataset requires the utilization of deeper networks to fully leverage the data (i.e., all three networks are remarkably deep). The size (i.e., in terms of the number of parameters it contains), the success rate (i.e., on the ImageNet), and the required training time for the applied networks are demonstrated (i.e., can be compared) in Figure 3.11.

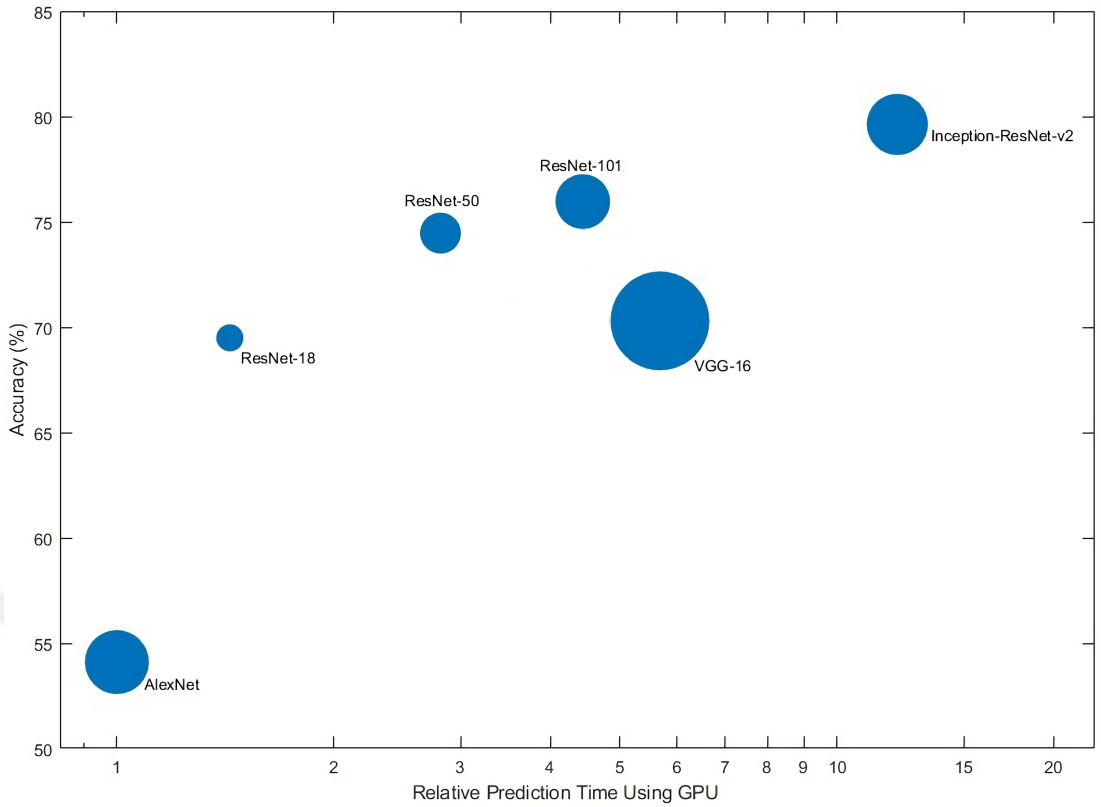


Figure 3.11 The correlation between the validation accuracy on ImageNet and the time required for prediction of tested NNs. The x-axis illustrates the relative prediction time, while the y-axis displays the validation accuracy. The size of the network is indicated by the blue circles [122]

Upon observing that the successes of AlexNet and VGG-16 were insufficient compared to ResNets on the dataset, the attention turned towards exploring various configurations of ResNets (i.e., to determine which network type and training parameters were the most effective ones). As the number of epochs increased, ResNet-18 showed a significant improvement in performance, as did ResNet-50. After some of these have been tried, the deeper models (i.e., ResNet-101 and Inception ResNet V2) have been inspected. Through fine-tuning the pre-processing and training phases, including optimizing image augmentation settings and determining optimal training epochs, the final results were obtained utilizing a standardized pre-processing stage (i.e., used for all trainings). Results for all these trials are presented in Table 4.1. Even the Inception ResNet V2 is the deepest one among other considered ResNets, ResNet-101 gave the best result, but they were very similar (i.e., may be due to not finding the best parameters). After that, the auto-cropped dataset was trained using the ResNet-101 (see Figure 3.12) as it was the most successful one for the non-auto-cropped dataset.

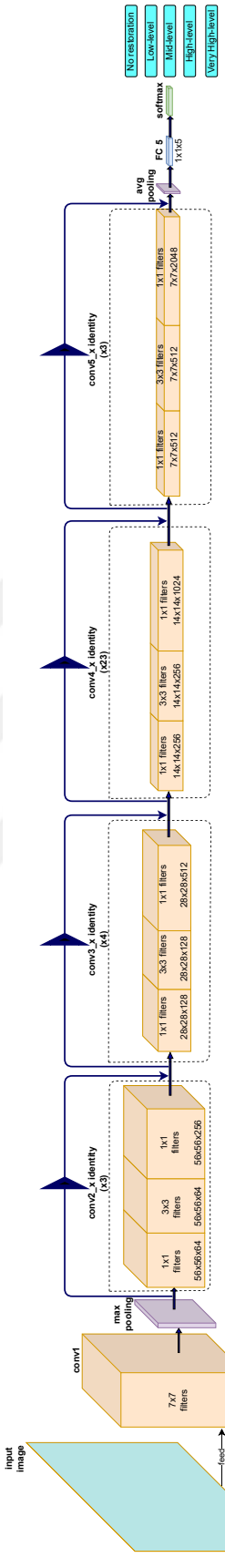


Figure 3.12 Representation of the structural design of ResNet-101 architecture, showcasing the various layers, components and connections that make up this powerful DNN [1]

Minor adjustments to the original network architectures were made, such as altering the input size from 3 channels to 1 channel (i.e., the dataset consists of grayscale images) and the resolution from 224×224 to 200×370, or modifying the size of the FC layer from 1000 to 5 (i.e., to facilitate classification of the 5 categories). As the source domain of pre-trained networks (i.e., ImageNet) consists of 3-channel images and is trained to classify objects (e.g., cat, minibus, or banana), where the dataset of the study consists of grayscale images and different types of teeth, it was inapplicable to utilize transfer learning. Consequently, all models were trained from scratch.

10-fold cross-validation, where the dataset is divided into 10 partitions and iteratively trained and validated on each partition to obtain a less optimistic and unbiased estimate of model performance, was employed. This involved separating 9 folds for training and 1 fold for validation and iterating this process 10 times, with the validation fold advancing to the next one each time. This enabled the generalization of the results and ensured reliable accuracy. The average accuracy of the 10 different trainings was calculated for evaluation. Figure 3.13 illustrates the 10-fold cross-validation process.

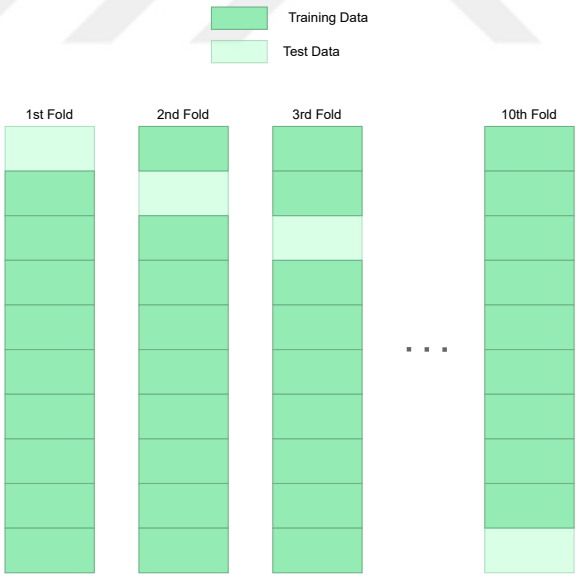


Figure 3.13 A visual representation of the 10-fold cross-validation approach

While training very deep NNs (i.e., having a huge number of parameters), the Video Random Access Memory (VRAM) of the GPU is under a heavy load, and if it is not sufficient, alternative actions must be taken. Accordingly, a mini-batch size of 24 was chosen in most trials. 36 and 48 options were also used for trouble-free models. The

'adam' solver was utilized for updating weights and biases. The learning rate was established as 1e-3. The determination of the number of epochs was primarily based on a trial-and-error methodology. However, after training was complete and results were available, it turned to increase the number of epochs until there was no further improvement in accuracy (i.e., until the model overfitted the data), then reducing the number of epochs to look for a better option somewhere in between.

CHAPTER 4

RESULTS AND DISCUSSION

In order to determine the optimal network configuration with the most appropriate pre-training and training parameters for the given dataset, a comprehensive examination of various trainings was conducted. Once the pre-training options were established as outlined in Section 3.1.1, they were fixed for all subsequent trainings. In each training session, multiple networks were evaluated using a various number of epochs.

The duration of training for each experiment was recorded (i.e., for 1-fold), and the top-1 accuracy results were presented in Table 4.1. Additionally, the confusion matrix for each model was also reported in Figure 4.1 (i.e., for the non-cropped dataset). This serves as a tabular representation of the model's performance, where each entry indicates the number of predictions made by the model. This can be used to calculate various performance metrics such as accuracy (i.e., the actual performance of the model), sensitivity (i.e., also called recall and specifies the ratio of correctly classified positive samples to all positive samples), precision (i.e., the ratio of predicted positive samples that are actually positive), and specificity (i.e., the ratio of correctly classified negative samples to all negative samples), which are calculated as follows:

$$Accuracy = \frac{TP + TN}{P + N} \tag{4.1}$$

$$Sensitivity\ (Recall) = \frac{TP}{TP + FN} \tag{4.2}$$

$$Precision = \frac{TP}{TP + FP} \tag{4.3}$$

$$Specificity = \frac{TN}{FP + TN} \tag{4.4}$$

It is important to note that in the context of evaluating the performance of a model, TN refers to true negative, TP refers to true positive, FN refers to false negative, FP refers to false positive, and P and N refer to positive and negative classifications, respectively.

In order to conduct a comprehensive comparison of the network performances, the Receiver Operating Characteristic (ROC) curve was calculated by varying the threshold for each class. Accordingly, each class's AUC value and their macro averages were taken into account. ROC analysis provides an insight into the ability of the model to separate samples of the positive class from the other classes.

Table 4.1 Accuracy results with their corresponding number of epochs and training durations. "Cropped v1" indicates using the auto-cropped dataset prepared using $max - min$ pooling in the pre-processing phase, while "Cropped v2" indicates using Sobel edge detector instead of $max - min$ pooling

Network Architecture	# of Epochs	Avg. Accuracy	Duration for 1 Fold
AlexNet	48	75.5%	297 min
AlexNet	128	68.6%	793 min
VGG-16	32	85.0%	1008 min
ResNet-18	36	90.0%	288 min
ResNet-18	48	90.9%	399 min
ResNet-18	72	91.6%	615 min
ResNet-18	100	92.0%	889 min
ResNet-18	120	92.1%	1030 min
ResNet-50	48	90.2%	1211 min
ResNet-50	72	91.7%	1789 min
ResNet-101	48	90.5%	1530 min
ResNet-101	72	91.8%	2210 min
ResNet-101	96	92.7%	3099 min
ResNet-101	112	92.2%	3638 min
ResNet-101 (cropped v1)	144	**94.5%**	5009 & 2663 min
ResNet-101 (cropped v2)	144	92.9%	4993 & 2681 min
Inception ResNet V2	64	92.1%	5349 min
Inception ResNet V2	96	91.7%	7699 min

All experiments were conducted on a computer equipped with an Intel® Core™ i7-5960X CPU, 32 GB of RAM, and an 8 GB NVIDIA® Quadro® M4000 GPU. It is important to note that while the hardware capabilities play a crucial role in reducing training time, they are not essential for practical usage in a clinical setting. In fact, even a GPU is not required, and today's average equipment is sufficient to perform a test on the trained network. It can be clearly said that the method is highly advantageous for clinical applications due to its high accuracy, robustness, and rapid execution time. MATLAB was used in all trainings except for the auto-cropped one. The auto-cropped dataset was trained on both Python and MATLAB, and their execution durations (i.e., the first one belongs to MATLAB and the second one

belongs to Python for the auto-cropped results) are presented in Table 4.1.

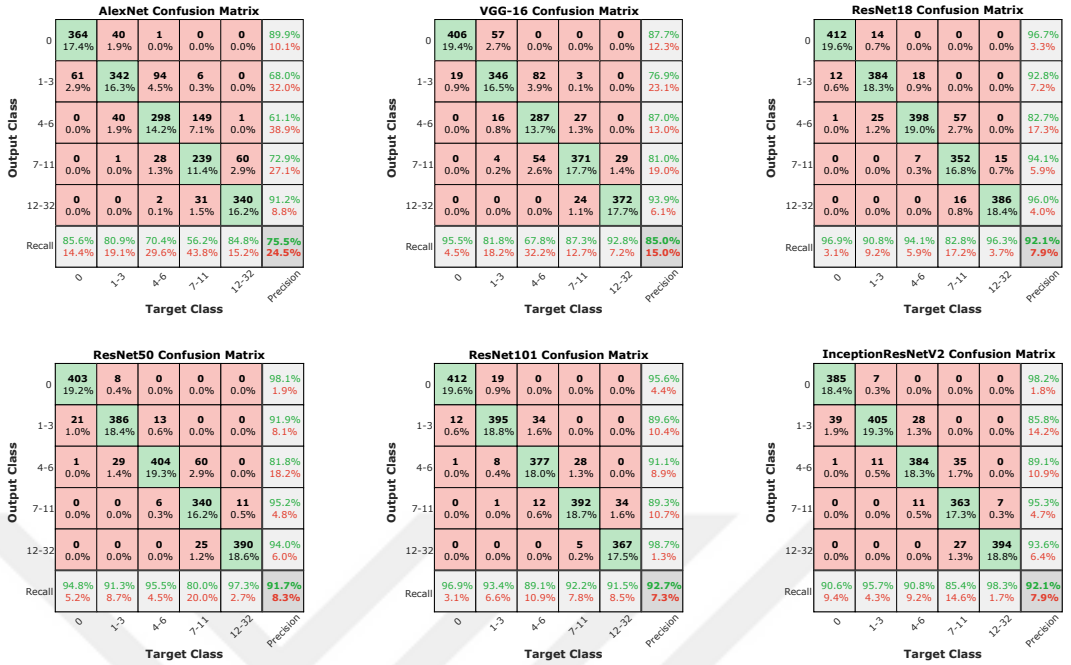


Figure 4.1 Confusion matrices for all tested models on the non-auto-cropped dataset, also feature recall (depicted in the bottom row) and precision (depicted in the rightmost column) values in a visual format. Rows represent the output class (i.e., predicted), while columns represent the target class (i.e., ground truth) [1]

The performance of all models was evaluated by utilizing confusion matrices, which are N×N matrices used for classification performance evaluation, where N represents the number of target classes. The confusion matrices, along with precision and recall values, and ROC curves for each class are depicted in Figure 4.1 and Figure 4.3, respectively. The confusion matrices for auto-cropped trainings are depicted in Figure 4.2.

The highest accuracy achieved for the non-auto-cropped dataset was 92.7% with ResNet-101, which was trained with 96 epochs. As the number of epochs increased for this network, the accuracy also increased until 96 epochs. Beyond this point, the model began to overfit the data, as evidenced by the decreased accuracy in the training with 112 epochs. The most successful training for the non-auto-cropped dataset also resulted in the highest macro-average AUC (i.e., the arithmetic mean of all AUC scores) at 0.989. The model obtained the best AUC value among all models for the 'no-restoration' and 'high-level-restoration' classes and the second-best AUC value for the 'low-level-restoration' class (see Figure 4.4). The macro-average AUC values

for all models are presented in Figure 4.5. Figure 4.3 and Figure 4.5 use a log-scale on the x-axis to show the difference in better terms.

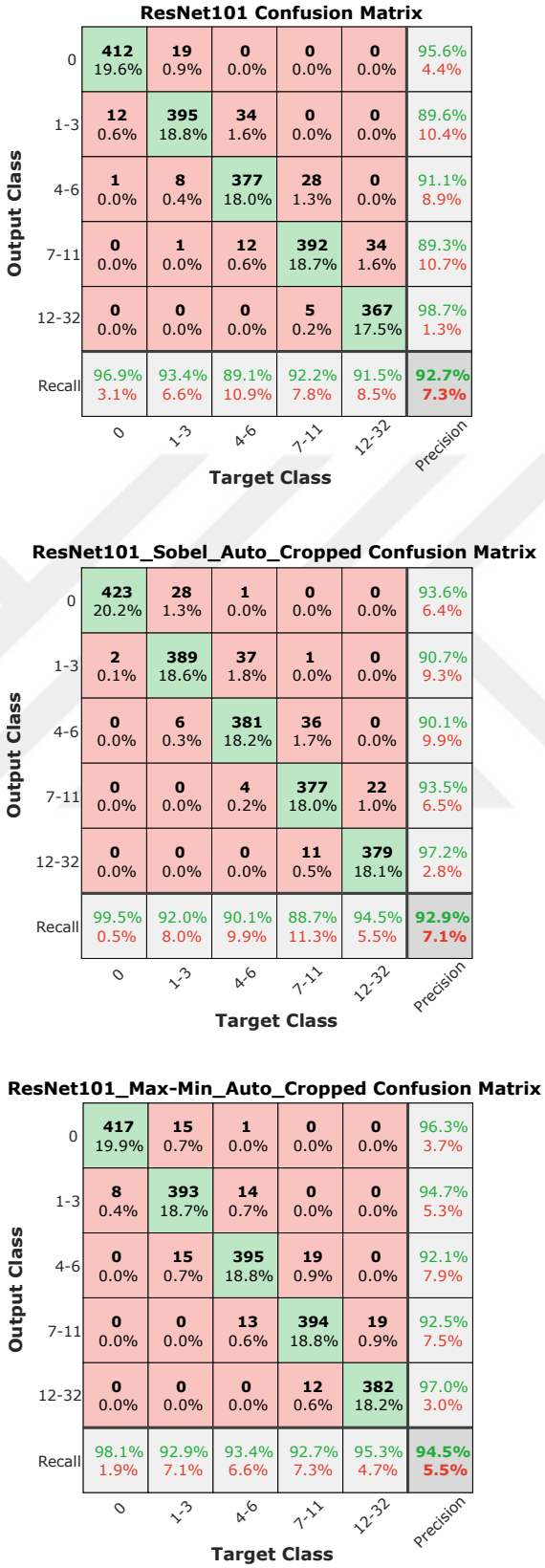


Figure 4.2 Confusion matrices for auto-cropped trainings and their comparison with the best of non-auto-cropped trainings

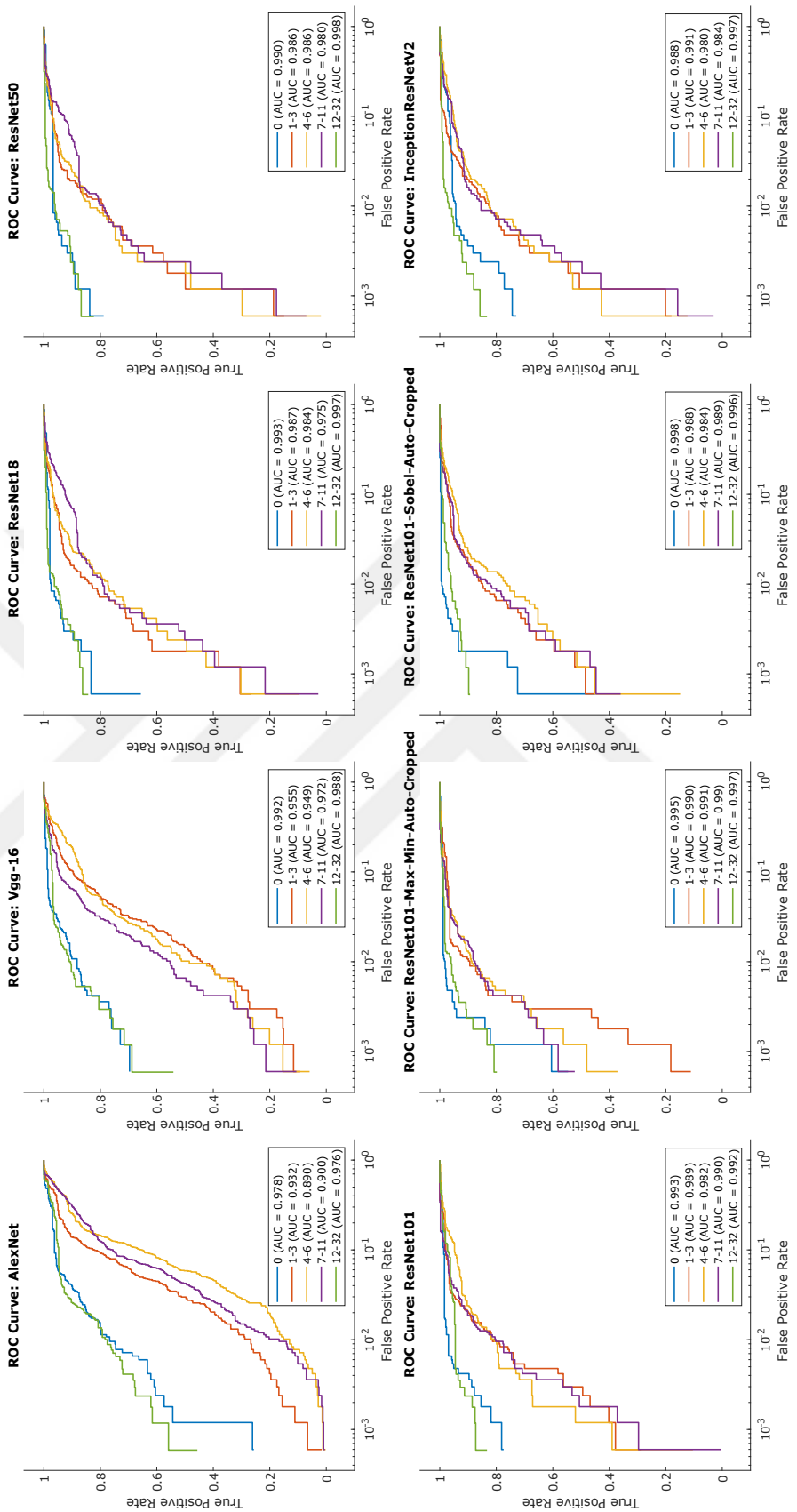


Figure 4.3 ROC curves, the models' ability to separate positive from negative samples, for all networks with the accompanying AUC values ranging from 0 to 1 (i.e., 1 indicates perfect class separation and 0.5 indicates no class separation) for each class [1]

In the non-auto-cropped dataset trials, the other highest accuracies obtained were 75.5% for AlexNet, 85.0% for VGG-16, 92.1% for ResNet-18, 91.7% for ResNet-50, and 92.1% for Inception ResNet V2. ResNet-101, the most successful network among others, had 92.7% accuracy. It can be concluded that all the tested networks performed well, with the ResNets displaying particularly impressive performance. Despite being different versions (i.e., with varying depths) of the same architecture, the accuracy, precision, recall, and specificity values of the ResNet models were similar. Additionally, none of the ResNet models displayed a clear dominance over the others in all target categories, as can be seen from the AUC values for each class in Figure 4.4. As a result, any of the ResNet models can be considered very successful in classifying the data. In general, ResNets have lower sensitivity scores in the 'high-level restoration' category but mostly exhibited high specificity scores (e.g., the lowest specificity score of ResNet101 is in the 'high-level restoration' category with 0.9719).

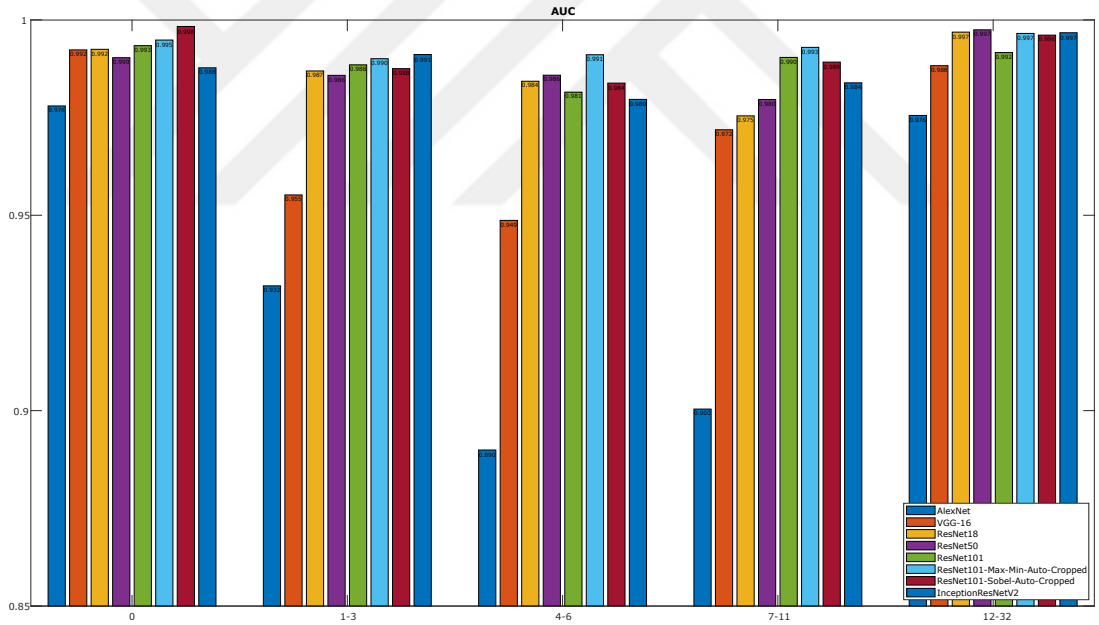


Figure 4.4 A bar comparison of the Area Under the Curve (AUC) values for all models in each class, where larger values indicate superior performance of the classifier [1]

When comparing the less successful networks, AlexNet and VGG-16, it can be observed that there are more noticeable differences in AUC values between classes. It appears that the classes 'no-restoration' and 'very-high-level restoration' were easier for the networks to learn, while the remaining classes 'low-level restoration', 'mid-level restoration', and 'high-level restoration' were more challenging for the less powerful networks. However, for the more powerful ResNet models, there was no significant difference observed

between classes.

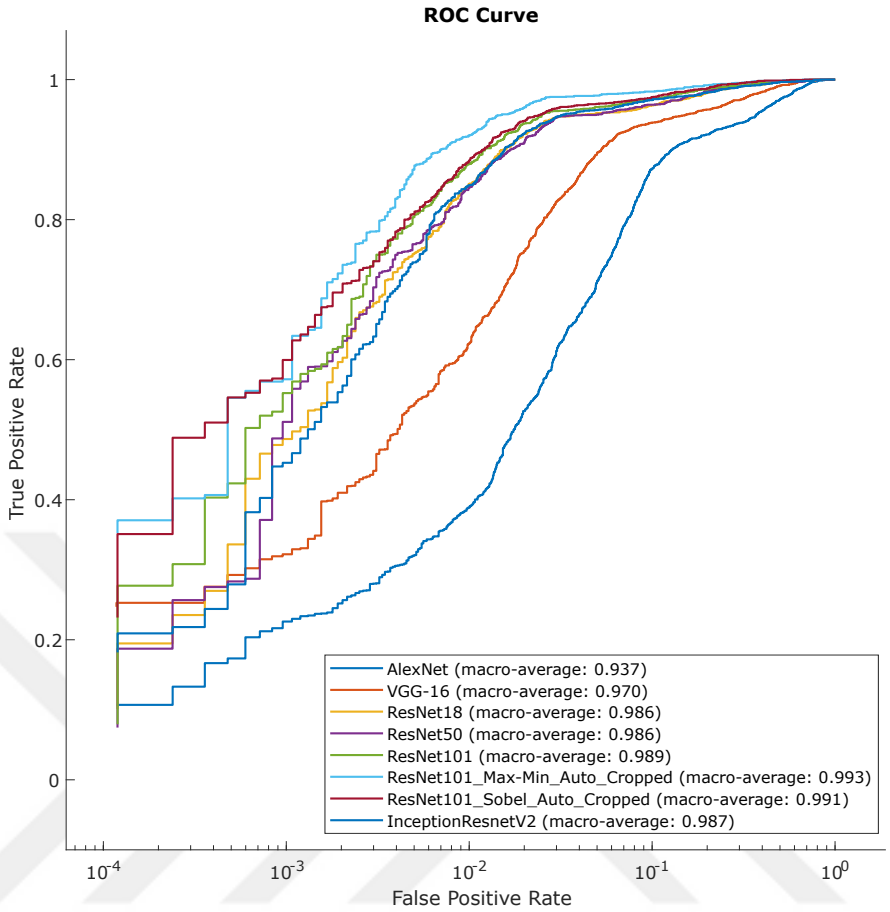


Figure 4.5 A graphical comparison of the macro-average AUC values for all networks

Auto-cropped (v1) training (i.e., the dataset prepared using $max-min$ pooling in the pre-processing phase) was the most successful one in almost all terms. It achieved 94.5% accuracy with a 0.993 macro-average AUC value. The model obtained the best AUC value among all models for the 'no restoration', 'mid-level restoration', 'high-level restoration', and 'very high-level restoration' classes, and the second-best AUC value for the 'low-level restoration' class (see Figure 4.4).

Auto-cropped (v2) training (i.e., the dataset prepared using Sobel edge detector instead of $max-min$ pooling) outperformed the best of non-auto-cropped trainings, with an accuracy of 92.9% and a 0.991 macro-average AUC value. The reason Sobel (v2 training) performed worse than $max-min$ pooling (v1 training) is that the resulting image had thinner edges than when using $max-min$ pooling (see Figure 3.7; using $max-min$ pooling provided edges in a blurrier and thicker way), as this leads to worse cropping results and hence worse accuracy. It is obvious that $max-min$ pooling would

be a popular choice for the DL community, as it is widely known in the community, but it also provided better cropping performance than Sobel.

4.1 Conclusion and Future Work

In this thesis, the aim was to attain a high success rate on a previously unstudied task utilizing a unique and private dataset through the application of DL methods. Through experimentation, the most favorable outcome was achieved by utilizing the auto-crop optimization with ResNet101 on the dataset. As demonstrated in Table 4.1, all tested networks, with the exception of AlexNet and VGG-16 (i.e., the least successful networks among others for another dataset, ImageNet, as can be seen in Figure 3.11), performed exceptionally well, surpassing 90% accuracy. This illustrates that DL methods are highly effective and are likely to continue to be so in the future. The key factor in achieving success in this effort was the dataset, as even poorly performing networks achieved over 80% accuracy, which can be considered a positive result.

The utilization of image augmentation was found to be beneficial not only for overall performance but also for reducing overfitting and determining appropriate parameters. It is highly recommended to use data augmentation even if the dataset is substantially big. Although the augmentation parameter limits were constrained for the non-auto-cropped dataset (i.e., it was an excessive augmentation), it caused some features to be lost. Smart cropping made it possible to get rid of noisy data, so extreme augmentation adjustment was no longer needed.

Auto-crop was also beneficial to take advantage of increasing the input size (i.e., 200×370 means increased resolution by 65.2%) of the network without corrupting the aspect ratio (i.e., earlier trainings were corrupting the aspect ratio when resizing to 224×224). The crop operation noticeably improved accuracy because it made it possible to focus on all necessary features while eliminating unnecessary ones, and it gave more stable results (i.e., without cropping, the random augmentation could choose to include noisy or unnecessary parts of the image). The CNN design can include auto-crop as a layer, notably after the input layer, to make it easier to crop and resize the network's input data. The Theseus layer [123] may be used to accomplish

this implementation, which permits training with backpropagation. This integration may be completed without the need for additional monitoring or adjustments to the optimization process. The auto-crop capability also permits the use of datasets with various resolutions, facilitating the processing of multi-resolution data.

The training process was found to be time-consuming, as 10-fold cross-validation (i.e., computationally expensive) was applied for each training session, resulting in a total training time that was 10 times the duration of a single fold, which is reported in Table 4.1. Additionally, fitting the training data and all network parameters into the VRAM of the GPU was a challenging task, often requiring a decrease in batch size (also limiting the input size). A more powerful GPU with greater VRAM would have been beneficial in these experiments.

The results of this study demonstrate that both the employed dataset and the methodology are powerful enough to achieve a high success rate, creating a useful tool for dental experts in providing initial information about restorations upon examination of a panoramic radiograph. Moreover, since the data set used in this thesis contains a sufficient number of examples, it is suitable for solving other problems. Also, the auto-crop algorithm is not a problem-specific solution but can be used for other grayscale datasets.

Future work includes the publication of the trained model (i.e., as a pre-trained model with the weights) for use in transfer learning in future studies, as well as making the dataset publicly available. Additionally, providing qualitative information about restorations, such as type, region, or material, is another potential avenue for future research on the dataset. Furthermore, re-labeling the entire dataset according to specific diagnostic needs is another potential area, as panoramic radiographs can be utilized for a variety of diagnoses. Apart from these, the auto-crop algorithm can be implemented as a trainable layer in an end-to-end NN because it is convenient as it is optimized with gradient ascent. Additionally, a better data augmentation structure (i.e., without having black regions) may also be achieved by locating RoI.

REFERENCES

[1] Top A. E., Ozdogan M. S., and Yeniad M., "Quantitative level determination of fixed restorations on panoramic radiographs using deep learning," *International Journal of Computerized Dentistry*, vol. 26, no. 3, 2023.

[2] Koza J. R., Bennett F. H., Andre D., and Keane M. A., "Automated design of both the topology and sizing of analog electrical circuits using genetic programming," in *Artificial Intelligence in Design'96*. Springer, 1996, pp. 151–170.

[3] LeCun Y., Boser B., Denker J. S., Henderson D., Howard R. E., Hubbard W., and Jackel L. D., "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[4] Zeiler M. D. and Fergus R., "Visualizing and understanding convolutional networks," in *European conference on computer vision*. Springer, 2014, pp. 818–833.

[5] Top A. E. and Kaya H., "Classification of eeg signals by using transfer learning on convolutional neural networks via spectrogram," in *ICENTE18*, 10 2018, pp. 137–142.

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

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

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

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

[10] He K., Zhang X., Ren S., and Sun J., "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[11] Russakovsky O., Deng J., Su H., Krause J., Satheesh S., Ma S., Huang Z., Karpathy A., Khosla A., Bernstein M. *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

[12] Top A. E., "Classification of eeg signals using transfer learning on convolutional neural networks via spectrogram," M.Sc. thesis, Ankara Yıldırım Beyazıt Üniversitesi Fen Bilimleri Enstitüsü, 2018.

[13] Griffin G., Holub A., and Perona P., "Caltech-256 object category dataset," *California Institute of Technology*, vol. 6, 2007.

[14] Angermueller C., Pärnamaa T., Parts L., and Stegle O., "Deep learning for computational biology," *Molecular systems biology*, vol. 12, no. 7, p. 878, 2016.

[15] Xie Q., Ding T., and Yang G., "Rehabilitation of oral function with removable dentures–still an option?" *Journal of Oral Rehabilitation*, vol. 42, no. 3, pp. 234–242, 2015.

[16] The American Board of Prosthodontics, "Examination guidelines for the certification process," accessed: 10-10-2020. [Online]. Available: https://www.abpros.org/application/files/5015/8946/1367/ABP-Guidelines-2020-01-20.pdf

[17] Zitzmann N. U., Hagmann E., and Weiger R., "What is the prevalence of various types of prosthetic dental restorations in europe?" *Clinical oral implants research*, vol. 18, pp. 20–33, 2007.

[18] Shamblott Family Dentistry, "Dental crowns," accessed: 10-09-2021. [Online]. Available: https://shamblottfamilydentistry.com/dental-crowns

[19] Palkalai Dental Clinic, "What are dental bridges?" accessed: 10-09-2021. [Online]. Available: https://shamblottfamilydentistry.com/dental-crowns

[20] Sakaguchi R. L. and Powers J. M., *Craig's restorative dental materials-e-book*. Elsevier Health Sciences, 2012.

[21] Haghanifar A., Majdabadi M. M., and Ko S.-B., "Paxnet: Dental caries detection in panoramic x-ray using ensemble transfer learning and capsule classifier," *arXiv preprint arXiv:2012.13666*, 2020.

[22] Shah N., Bansal N., and Logani A., "Recent advances in imaging technologies in dentistry," *World journal of radiology*, vol. 6, no. 10, p. 794, 2014.

[23] Carneiro J., Caldas I., Afonso A., and Cardoso H., "Examining the socioeconomic effects on third molar maturation in a portuguese sample of children, adolescents and young adults," *International journal of legal medicine*, vol. 131, no. 1, pp. 235–242, 2017.

[24] Fitzgerald R., "Phase-sensitive x-ray imaging," *Physics today*, vol. 53, no. 7, pp. 23–26, 2000.

[25] Pröbster L. and Diehl J., "Slip-casting alumina ceramics for crown and bridge restorations." *Quintessence International*, vol. 23, no. 1, 1992.

[26] Keerthna M. and Jain A. R., "Comparison of accuracy of digital radiography and panoramic radiography in dental implants procedure-a literature review." *Drug Invention Today*, vol. 10, no. 5, 2018.

[27] Miyazaki T. and Hotta Y., "Cad/cam systems available for the fabrication of crown and bridge restorations," *Australian dental journal*, vol. 56, pp. 97–106, 2011.

[28] Huumonen S. and Ørstavik D., "Radiological aspects of apical periodontitis," *Endodontic Topics*, vol. 1, no. 1, pp. 3–25, 2002.

[29] White S. C., Heslop E. W., Hollender L. G., Mosier K. M., Ruprecht A., Shrout M. K. *et al.*, "Parameters of radiologic care: An official report of the american academy of oral and maxillofacial radiology," *Oral Surgery, Oral Medicine, Oral Pathology, Oral Radiology, and Endodontology*, vol. 91, no. 5, pp. 498–511, 2001.

[30] Liedke G. S., Spin-Neto R., da Silveira H., and Wenzel A., "Radiographic diagnosis of dental restoration misfit: a systematic review," *Journal of oral rehabilitation*, vol. 41, no. 12, pp. 957–967, 2014.

[31] Liedke G. S., Spin-Neto R., Vizzotto M. B., Da Silveira P. F., Silveira H. E. D., and Wenzel A., "Diagnostic accuracy of conventional and digital radiography for detecting misfit between the tooth and restoration in metal-restored teeth," *The Journal of prosthetic dentistry*, vol. 113, no. 1, pp. 39–47, 2015.

[32] Papageorgiou S., Papadelli A., Koidis P., and Petridis H., "The effect of prosthetic margin location on caries susceptibility. a systematic review and meta-analysis," *British dental journal*, vol. 214, no. 12, pp. 617–624, 2013.

[33] Scarfe W. C. and Farman A. G., "What is cone-beam ct and how does it work?" *Dental Clinics of North America*, vol. 52, no. 4, pp. 707–730, 2008.

[34] Anbiaee N., Mohassel A., Imanimoghaddam M., and Moazzami S., "A comparison of the accuracy of digital and conventional radiography in the diagnosis of recurrent caries," *J Contemp Dent Pract*, vol. 11, no. 6, pp. E025–32, 2010.

[35] Corbet E., Ho D., and Lai S., "Radiographs in periodontal disease diagnosis and management," *Australian dental journal*, vol. 54, pp. S27–S43, 2009.

[36] Rushton V., Horner K., and Worthington H., "Screening panoramic radiology of adults in general dental practice: radiological findings," *British dental journal*, vol. 190, no. 9, pp. 495–501, 2001.

[37] Flygare L. and Öhman A., "Preoperative imaging procedures for lower wisdom teeth removal," *Clinical oral investigations*, vol. 12, no. 4, pp. 291–302, 2008.

[38] Freeman J. P. and Brand J. W., "Radiation doses of commonly used dental radiographic surveys," *Oral surgery, oral medicine, oral pathology*, vol. 77, no. 3, pp. 285–289, 1994.

[39] Chan H.-L., Misch K., and Wang H.-L., "Dental imaging in implant treatment planning," *Implant dentistry*, vol. 19, no. 4, pp. 288–298, 2010.

[40] Choi J.-W., "Assessment of panoramic radiography as a national oral examination tool: review of the literature," *Imaging science in dentistry*, vol. 41, no. 1, pp. 1–6, 2011.

[41] Jader G., Fontineli J., Ruiz M., Abdalla K., Pithon M., and Oliveira L., "Deep instance segmentation of teeth in panoramic x-ray images," in *2018 31st SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*. IEEE, 2018, pp. 400–407.

[42] Caldeira M., Serrano A., Quaresma R., Pedron C., and Romão M., "Information and communication technology adoption for business benefits: A case analysis of an integrated paperless system," *International Journal of Information Management*, vol. 32, no. 2, pp. 196–202, 2012.

[43] Rajpurkar P., Irvin J., Zhu K., Yang B., Mehta H., Duan T., Ding D., Bagul A., Langlotz C., Shpanskaya K. *et al.*, "Chexnet: Radiologist-level pneumonia detection on chest x-rays with deep learning," *arXiv preprint arXiv:1711.05225*, 2017.

[44] Yeshua T., Mandelbaum Y., Abdalla-Aslan R., Nadler C., Cohen L., Zemour L., Kabla D., Gleisner O., and Leichter I., "Automatic detection and classification of dental restorations in panoramic radiographs," *Issues in Informing Science and Information Technology*, vol. 16, pp. 221–234, 2019.

[45] Nelson R. R., Buterbaugh K., Perl M., and Gelijns A., "How medical know-how progresses," *Research policy*, vol. 40, no. 10, pp. 1339–1344, 2011.

[46] Sathya R. and Saleena B., "A survey on content based image retrieval using convolutional neural networks," *International Journal*, vol. 9, no. 5, 2020.

[47] Sklan J. E., Plassard A. J., Fabbri D., and Landman B. A., "Toward content-based image retrieval with deep convolutional neural networks," in *Medical Imaging 2015: Biomedical Applications in Molecular, Structural, and Functional Imaging*, vol. 9417. International Society for Optics and Photonics, 2015, p. 94172C.

[48] Doi K., "Computer-aided diagnosis in medical imaging: historical review, current status and future potential," *Computerized medical imaging and graphics*, vol. 31, no. 4-5, pp. 198–211, 2007.

[49] Alharbi A., Alharbi S., and Alqaidi S., "Guidelines for dental care provision during the covid-19 pandemic," *The Saudi Dental Journal*, 2020.

[50] Lee J.-H., Kim D.-H., Jeong S.-N., and Choi S.-H., "Detection and diagnosis of dental caries using a deep learning-based convolutional neural network algorithm," *Journal of dentistry*, vol. 77, pp. 106–111, 2018.

[51] Çelik B. and Çelik M. E., "Automated detection of dental restorations using deep learning on panoramic radiographs," *Dentomaxillofacial Radiology*, vol. 51, no. 8, p. 20220244, 2022.

[52] Heo M.-S., Kim J.-E., Hwang J.-J., Han S.-S., Kim J.-S., Yi W.-J., and Park I.-W., "Artificial intelligence in oral and maxillofacial radiology: what is currently possible?" *Dentomaxillofacial Radiology*, vol. 50, no. 3, p. 20200375, 2021.

[53] Kwon O., Yong T.-H., Kang S.-R., Kim J.-E., Huh K.-H., Heo M.-S., Lee S.-S., Choi S.-C., and Yi W.-J., "Automatic diagnosis for cysts and tumors of both jaws on panoramic radiographs using a deep convolution neural network," *Dentomaxillofacial Radiology*, vol. 49, no. 8, p. 20200185, 2020.

[54] Hung K., Montalvao C., Tanaka R., Kawai T., and Bornstein M. M., "The use and performance of artificial intelligence applications in dental and maxillofacial radiology: A systematic review," *Dentomaxillofacial Radiology*, vol. 49, no. 1, p. 20190107, 2020.

[55] Wang C.-W., Huang C.-T., Lee J.-H., Li C.-H., Chang S.-W., Siao M.-J., Lai T.-M., Ibragimov B., Vrtovec T., Ronneberger O. *et al.*, "A benchmark for comparison of dental radiography analysis algorithms," *Medical image analysis*, vol. 31, pp. 63–76, 2016.

[56] Yang J., Xie Y., Liu L., Xia B., Cao Z., and Guo C., "Automated dental image analysis by deep learning on small dataset," in *2018 IEEE 42nd Annual Computer*

Software and Applications Conference (COMPSAC), vol. 1. IEEE, 2018, pp. 492–497.

[57] Prajapati S. A., Nagaraj R., and Mitra S., "Classification of dental diseases using cnn and transfer learning," in *2017 5th International Symposium on Computational and Business Intelligence (ISCBI)*. IEEE, 2017, pp. 70–74.

[58] Miki Y., Muramatsu C., Hayashi T., Zhou X., Hara T., Katsumata A., and Fujita H., "Classification of teeth in cone-beam ct using deep convolutional neural network," *Computers in biology and medicine*, vol. 80, pp. 24–29, 2017.

[59] Srivastava M. M., Kumar P., Pradhan L., and Varadarajan S., "Detection of tooth caries in bitewing radiographs using deep learning," *arXiv preprint arXiv:1711.07312*, 2017.

[60] Lee J.-H., Kim D.-h., Jeong S.-N., and Choi S.-H., "Diagnosis and prediction of periodontally compromised teeth using a deep learning-based convolutional neural network algorithm," *Journal of periodontal & implant science*, vol. 48, no. 2, pp. 114–123, 2018.

[61] Thanathornwong B. and Suebnukarn S., "Automatic detection of periodontal compromised teeth in digital panoramic radiographs using faster regional convolutional neural networks," *Imaging Science in Dentistry*, vol. 50, no. 2, p. 169, 2020.

[62] Sukegawa S., Yoshii K., Hara T., Yamashita K., Nakano K., Yamamoto N., Nagatsuka H., and Furuki Y., "Deep neural networks for dental implant system classification," *Biomolecules*, vol. 10, no. 7, p. 984, 2020.

[63] Li Z., Wang S.-H., Fan R.-R., Cao G., Zhang Y.-D., and Guo T., "Teeth category classification via seven-layer deep convolutional neural network with max pooling and global average pooling," *International Journal of Imaging Systems and Technology*, vol. 29, no. 4, pp. 577–583, 2019.

[64] Top A. E., Torun F. Ş., and Kaya H., "Parallel and distributed image segmentation based on colors using k-means clustering algorithm," in *Proceedings of the ICES 2019: 5th International Conference on Engineering Sciences*, 2019.

[65] Top A. E., Torun F. Ş., and Kaya H., "Parallel k-means clustering with naïve sharding for unsupervised image segmentation via mpi," *Journal of Engineering Sciences and Design*, vol. 8, no. 3, pp. 791–798, sep 2020. [Online]. Available: https://doi.org/10.21923/jesd.748209

[66] Silva G., Oliveira L., and Pithon M., "Automatic segmenting teeth in x-ray images: Trends, a novel data set, benchmarking and future perspectives," *Expert Systems with Applications*, vol. 107, pp. 15–31, 2018.

[67] Chen H., Zhang K., Lyu P., Li H., Zhang L., Wu J., and Lee C.-H., "A deep learning approach to automatic teeth detection and numbering based on object detection in dental periapical films," *Scientific reports*, vol. 9, no. 1, pp. 1–11, 2019.

[68] Tuzoff D. V., Tuzova L. N., Bornstein M. M., Krasnov A. S., Kharchenko M. A., Nikolenko S. I., Sveshnikov M. M., and Bednenko G. B., "Tooth detection and numbering in panoramic radiographs using convolutional neural networks," *Dentomaxillofacial Radiology*, vol. 48, no. 4, p. 20180051, 2019.

[69] Tian S., Dai N., Zhang B., Yuan F., Yu Q., and Cheng X., "Automatic classification and segmentation of teeth on 3d dental model using hierarchical deep learning networks," *IEEE Access*, vol. 7, pp. 84 817–84 828, 2019.

[70] Jaskari J., Sahlsten J., Järnstedt J., Mehtonen H., Karhu K., Sundqvist O., Hietanen A., Varjonen V., Mattila V., and Kaski K., "Deep learning method for mandibular canal segmentation in dental cone beam computed tomography volumes," *Scientific reports*, vol. 10, no. 1, pp. 1–8, 2020.

[71] Kim J., Lee H.-S., Song I.-S., and Jung K.-H., "Dentnet: Deep neural transfer network for the detection of periodontal bone loss using panoramic dental radiographs," *Scientific reports*, vol. 9, no. 1, pp. 1–9, 2019.

[72] Imangaliyev S., van der Veen M. H., Volgenant C. M., Keijser B. J., Crielaard W., and Levin E., "Deep learning for classification of dental plaque images," in *International Workshop on Machine Learning, Optimization, and Big Data*. Springer, 2016, pp. 407–410.

[73] Abdalla-Aslan R., Yeshua T., Kabla D., Leichter I., and Nadler C., "An artificial intelligence system using machine-learning for automatic detection and classification of dental restorations in panoramic radiography: Automated detection and classification of panoramic dental restoration," *Oral Surgery, Oral Medicine, Oral Pathology and Oral Radiology*, 2020.

[74] Karatas O., Cakir N. N., Ozsariyildiz S. S., Kis H. C., Demirbuga S., and Gurgan C. A., "A deep learning approach to dental restoration classification from bitewing and periapical radiographs," *Quintessence Int*, vol. 52, pp. 568–74, 2021.

[75] Chen J., Bai G., Liang S., and Li Z., "Automatic image cropping: A computational complexity study," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 507–515.

[76] Liang Y., Lv J., Li D., Yang X., Wang Z., and Li Q., "Accurate cobb angle estimation on scoliosis x-ray images via deeply-coupled two-stage network with differentiable cropping and random perturbation," *IEEE Journal of Biomedical and Health Informatics*, vol. 27, no. 3, pp. 1488–1499, 2022.

[77] Riad R., Teboul O., Grangier D., and Zeghidour N., "Learning strides in convolutional neural networks," *arXiv preprint arXiv:2202.01653*, 2022.

[78] Rippel O., Snoek J., and Adams R. P., "Spectral representations for convolutional neural networks," *Advances in neural information processing systems*, vol. 28, 2015.

[79] Dai J., He K., and Sun J., "Instance-aware semantic segmentation via multi-task network cascades," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 3150–3158.

[80] Han C., Ye J., Zhong Y., Tan X., Zhang C., Gao C., and Sang N., "Re-id driven localization refinement for person search," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 9814–9823.

[81] Recasens A., Kellnhofer P., Stent S., Matusik W., and Torralba A., "Learning to zoom: a saliency-based sampling layer for neural networks," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 51–66.

[82] Jaderberg M., Simonyan K., Zisserman A. *et al.*, "Spatial transformer networks," *Advances in neural information processing systems*, vol. 28, 2015.

[83] He K., Gkioxari G., Dollár P., and Girshick R., "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[84] Girshick R., "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[85] Ren S., He K., Girshick R., and Sun J., "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, 2015.

[86] Dai J., Li Y., He K., and Sun J., "R-fcn: Object detection via region-based fully convolutional networks," *Advances in neural information processing systems*, vol. 29, 2016.

[87] Jiang W., Sun W., Tagliasacchi A., Trulls E., and Yi K. M., "Linearized multi-sampling for differentiable image transformation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 2988–2997.

[88] Mitchell T. M., *The discipline of machine learning*. Carnegie Mellon University, School of Computer Science, Machine Learning . . . , 2006, vol. 9.

[89] McCulloch W. S. and Pitts W., "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[90] Rosenblatt F., "The perceptron: a probabilistic model for information storage and organization in the brain." *Psychological review*, vol. 65, no. 6, p. 386, 1958.

[91] Werbos P., "Beyond regression:" new tools for prediction and analysis in the behavioral sciences," *Ph. D. dissertation, Harvard University*, 1974.

[92] Goodfellow I., Bengio Y., and Courville A., *Deep learning*. MIT press, 2016.

[93] Assael Y. M., Shillingford B., Whiteson S., and De Freitas N., "Lipnet: End-to-end sentence-level lipreading," *arXiv preprint arXiv:1611.01599*, 2016.

[94] Silver D., Huang A., Maddison C. J., Guez A., Sifre L., Van Den Driessche G., Schrittwieser J., Antonoglou I., Panneershelvam V., Lanctot M. *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[95] Vinyals O., Babuschkin I., Czarnecki W. M., Mathieu M., Dudzik A., Chung J., Choi D. H., Powell R., Ewalds T., Georgiev P. *et al.*, "Grandmaster level in starcraft ii using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, 2019.

[96] Zhou W., Yang Y., Yu C., Liu J., Duan X., Weng Z., Chen D., Liang Q., Fang Q., Zhou J. *et al.*, "Ensembled deep learning model outperforms human experts in diagnosing biliary atresia from sonographic gallbladder images," *Nature communications*, vol. 12, no. 1, pp. 1–14, 2021.

[97] Dechter R., "Learning while searching in constraint-satisfaction problems," in *Association for the Advancement of Artificial Intelligence-86 Proceedings*. AAAI, 1986.

[98] Huang J., "Accelerating ai with gpus: A new computing model," [Online]., 2016, [Accessed: 10-01-2023]. [Online]. Available: https://blogs.nvidia.com/blog/2016/01/12/accelerating-ai-artificial-intelligence-gpus/

[99] Schmidhuber J., "Deep learning in neural networks: An overview," *Neural networks*, vol. 61, pp. 85–117, 2015.

[100] Bengio Y. *et al.*, "Learning deep architectures for ai," *Foundations and trends® in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[101] Fukushima K., "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biological Cybernetics*, vol. 36, no. 4, pp. 193–202, 1980.

[102] Hubel D. H. and Wiesel T. N., "Receptive fields of single neurones in the cat's striate cortex," *The Journal of physiology*, vol. 148, no. 3, p. 574, 1959.

[103] LeCun Y., Bottou L., Bengio Y., and Haffner P., "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[104] Ranzato M., Poultney C., Chopra S., and Cun Y., "Efficient learning of sparse representations with an energy-based model," *Advances in neural information processing systems*, vol. 19, 2006.

[105] Chellapilla K., Puri S., and Simard P., "High performance convolutional neural networks for document processing," in *Tenth international workshop on frontiers in handwriting recognition*. Suvisoft, 2006.

[106] Bengio Y., Lamblin P., Popovici D., and Larochelle H., "Greedy layer-wise training of deep networks," *Advances in neural information processing systems*, vol. 19, 2006.

[107] Hinton G. E., Osindero S., and Teh Y.-W., "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[108] Ciresan D. C., Meier U., Masci J., Gambardella L. M., and Schmidhuber J., "Flexible, high performance convolutional neural networks for image classification," in *Twenty-second international joint conference on artificial intelligence*, 2011.

[109] Ciregan D., Meier U., and Schmidhuber J., "Multi-column deep neural networks for image classification," in *2012 IEEE conference on computer vision and pattern recognition*. IEEE, 2012, pp. 3642–3649.

[110] Abbas W. and Masip Rodo D., "Computer methods for automatic locomotion and gesture tracking in mice and small animals for neuroscience applications: a survey," *Sensors*, vol. 19, no. 15, p. 3274, 2019.

[111] Deshpande A., "A beginner's guide to understanding convolutional neural networks," [Online]., 2016, [Accessed:

10-01-2023]. [Online]. Available: https://adeshpande3.github.io/A-Beginner's-Guide-To-Understanding-Convolutional-Neural-Networks/

[112] Girshick R., Donahue J., Darrell T., and Malik J., "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[113] Chen Z., Cen J., and Xiong J., "Rolling bearing fault diagnosis using time-frequency analysis and deep transfer convolutional neural network," *Ieee Access*, vol. 8, pp. 150 248–150 261, 2020.

[114] Ioffe S. and Szegedy C., "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*. pmlr, 2015, pp. 448–456.

[115] Matsuyama E. *et al.*, "A deep learning interpretable model for novel coronavirus disease (covid-19) screening with chest ct images," *Journal of Biomedical Science and Engineering*, vol. 13, no. 07, p. 140, 2020.

[116] Mahtani A. A. and Pradeep S., "Radiographic analysis of various restorations on teeth–an in vitro study." *Drug Invention Today*, vol. 11, no. 8, 2019.

[117] Liu S., Lu Y., Wang J., Hu S., Zhao J., and Zhu Z., "A new focus evaluation operator based on max–min filter and its application in high quality multi-focus image fusion," *Multidimensional Systems and Signal Processing*, vol. 31, pp. 569–590, 2020.

[118] Paszke A., Gross S., Massa F., Lerer A., Bradbury J., Chanan G., Killeen T., Lin Z., Gimelshein N., Antiga L. *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.

[119] Krizhevsky A., Hinton G. *et al.*, "Learning multiple layers of features from tiny images," *University of Toronto*, 2009.

[120] Szegedy C., Ioffe S., Vanhoucke V., and Alemi A. A., "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 2017, pp. 4278–4284.

[121] Kolesnikov A., Beyer L., Zhai X., Puigcerver J., Yung J., Gelly S., and Houlsby N., "Big transfer (bit): General visual representation learning," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part V 16*. Springer, 2020, pp. 491–507.

[122] MathWorks, "Pretrained deep neural networks," [Online]., 2021, [Accessed: 10-03-2022]. [Online]. Available: https://www.mathworks.com/help/deeplearning/ug/pretrained-convolutional-neural-networks.html

[123] Pineda L., Fan T., Monge M., Venkataraman S., Sodhi P., Chen R. T., Ortiz J., DeTone D., Wang A., Anderson S. *et al.*, "Theseus: A library for differentiable nonlinear optimization," *Advances in Neural Information Processing Systems*, vol. 35, pp. 3801–3818, 2022.