ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL

A HYBRID DEEP LEARNING METAHEURISTIC MODEL FOR DIAGNOSIS OF DIABETIC RETINOPATHY

Ph.D. THESIS

Ömer Faruk GÜRCAN

Department of Industrial Engineering

Industrial Engineering Programme

OCTOBER 2022

ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL

A HYBRID DEEP LEARNING METAHEURISTIC MODEL FOR DIAGNOSIS OF DIABETIC RETINOPATHY

Ph.D. THESIS

Ömer Faruk GÜRCAN
(507142109)

Department of Industrial Engineering

Industrial Engineering Programme

Thesis Advisor: Assis. Prof. Dr. Ömer Faruk BEYCA

OCTOBER 2022

ISTANBUL TEKNİK ÜNİVERSİTESİ ★ LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

DİYABETİK RETİNOPATİNİN TANISI İÇİN HİBRİT BİR DERİN ÖĞRENME META-SEZGİSEL MODELİ

DOKTORA TEZİ

Ömer Faruk GÜRCAN
(507142109)

Endüstri Mühendisliği Anabilim Dalı

Endüstri Mühendisliği Programı

Tez Danışmanı: Dr. Öğr. Üyesi Ömer Faruk BEYCA

EKİM 2022

Ömer Faruk GÜRCAN, a Ph.D. student of ITU Graduate School student ID 507142109, successfully defended the dissertation entitled "A HYBRID DEEP LEARNING METAHEURISTIC MODEL FOR DIAGNOSIS OF DIABETIC RETINOPATHY", which he prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor :	**Assis. Prof. Dr. Ömer Faruk BEYCA** Istanbul Technical University	.....
Jury Members :	**Prof. Dr. Selim ZAİM** Istanbul Sabahattin Zaim University	.....
	Prof. Dr. Alp ÜSTÜNDAĞ Istanbul Technical University	.....
	Prof. Dr. Nizamettin BAYYURT Istanbul Technical University	.....
	Assis. Prof. Dr. Fuat KOSANOĞLU Yalova University	.....

Date of Submission : 12 September 2022
Date of Defense : 17 October 2022

To my parents, my dear wife, and sister, for their encouragement, patience, love, and prayers
Also to my dear daughter, Zeynep

FOREWORD

I would like to thank my supervisor, Asst. Prof. Ömer Faruk BEYCA, for his invaluable support, advice, and suggestions throughout all the phases of the dissertation.

Special thanks to Prof. Dr. Selim ZAİM and Prof. Dr. Alp ÜSTÜNDAĞ, for their guidance and support.

Special thanks to Dr. Uğur ATİCİ, for his help and suggestions.

Special thanks to my parents, my dear wife Derya, my dear daughter, and sister, for their love, encouragement, patience, and support.

October 2022

Ömer Faruk GÜRCAN
(Industrial Engineer)

TABLE OF CONTENTS

Page

ABBREVIATIONS

DR **:** Diabetic Retinopathy
CNN **:** Convolutional Neural Network
XGBoost **:** Extreme Gradient Boosting
NP **:** Nondeterministic Polynomial
ETDRS **:** Early Treatment Diabetic Retinopathy Study
NPDR **:** Non-Proliferative Diabetic Retinopathy
PDR **:** Proliferative Diabetic Retinopathy
ICDR **:** International Clinical Diabetic Retinopathy
CAD **:** Computer-aided diagnosis
ReLU **:** Rectified linear unit
ANN **:** Artificial Neural Network
NN **:** Neural Network
ML **:** Machine Learning
DL : Deep Learning
FCL : Fully connected layer
2D : Two-dimensional
Tanh : Hyperbolic tangent
Acc : Accuracy
SE : Sensitivity
SP : Specificity
Rec : Recall
PR : Precision
AUC : Area under the curve
PRC : Precision and Recall Curves
MHA : Metaheuristic Algorithm
RGB : Red green blue
ILSVRC : ImageNet large-scale visual recognition challenge
kNN : k-nearest neighbors
SVM : Support Vector Machine
PSO : Particle Swarm Optimization
SimAn : Simulated Annealing
ABC : Artificial Bee Colony
LR : Logistic Regression
SVC : Support Vector Classifier
MLP : Multilayer Perceptron
GAP : Global Average Pooling
Conv2d : Two-dimensional convolution layer
tp : true positive
tn : true negative
fp : false positive
fn : false negative
NPV : negative predictive value

LIST OF TABLES

Page

LIST OF FIGURES

Page

A HYBRID DEEP LEARNING METAHEURISTIC MODEL FOR DIAGNOSIS OF DIABETIC RETINOPATHY

SUMMARY

Diabetes is a disease that results in an increase in blood sugar due to the pancreas not producing enough insulin, insufficient effect of the produced insulin, or ineffective use of insulin. According to the International Diabetes Federation 2021 report, approximately 537 million adults aged between 20 and 79 live with diabetes worldwide. It is estimated that the number of people with diabetes will increase to 643 million in 2030 and 783 million in 2045.

Diabetic retinopathy (DR) is an eye condition that can cause vision loss, irrecoverable visual deterioration, and blindness in people with diabetes. Today, it is one of the leading diseases that cause blindness. Anyone with any diabetes can become a DR. In ophthalmology, type 2 diabetes can lead to DR if left untreated for more than five years. Diabetes-related high blood sugar leads to DR. Over time, having too much sugar in the blood damages the retina. The deterioration of this disease in the eye begins when sugar blocks the capillaries leading to the retina, causing fluid leakage or bleeding at a later stage. The eye produces new vessels to compensate for the blocked vessels, but these newly formed vessels often do not work well and can bleed or leak easily. DR can lead to other serious eye conditions. For example, about one in 15 people with diabetes develop diabetic macular edema over time. DR can lead to the formation of abnormal blood vessels in the retina and prevent fluid from leaving the eye. That causes a type of glaucoma.

It is crucial for people with diabetes to have a comprehensive eye examination at least once a year. Follow-up of diabetes; factors such as staying physically active, eating a healthy diet, and using medications regularly can stop the damage to the eye and help prevent or delay vision loss. Some risk factors increase the development of DR, such as pregnancy, uncontrolled diabetes, smoking addiction, hypertension, and high cholesterol.

In addition to being detected by magnifying the pupil in eye examination, DR is also diagnosed with the help of image processing techniques. It is common to use fundus images obtained by fundus fluorescent angiography to detect DR and other retinal diseases. Nowadays, with the increasing number of patients and the developments in imaging technologies, disease detection from medical images by various methods has increased.

Deep learning is one of the methods whose application area has increased exponentially in recent years. Deep learning is a subfield of machine learning; both are a subfield in artificial intelligence. Deep learning methods draw attention with their versatility, high performance, high generalization capacity, and multidisciplinary use. Technological developments such as the collection of large amounts of data, graphics processing units, the development of robust computer infrastructures, and cloud computing support the building and implementation of new models.

Increasing the number of images for a particular patient case and high-resolution images increases specialists' workload. Diagnosis of DR manually by an ophthalmologist is an expensive and time-consuming process. It requires experts who have remarkable experience. In addition, the complexity of medical images and the variations between specialists make it difficult for radiologists and physicians to make efficient and accurate diagnoses at any time. Deep learning is promising in providing decision support to clinicians by increasing the accuracy and efficiency of diagnosis and treatment processes of various diseases. Today, in some medical studies, the success levels of expert radiologists have been achieved or exceeded.

Convolutional neural networks (CNNs) are the most widely used deep learning networks in image recognition, image/object recognition, or classification studies. A CNN model doesn't need manually designed features for training; it extracts features from data directly while network training on images. The automated feature extraction property and their success make CNNs highly preferred models in computer vision tasks.

This study proposes a hybrid model for the automatic diagnosis of DR. A binary classification of DR (referable vs. non-referable DR) is made using a deep CNN model, metaheuristic algorithms, and machine learning algorithms. A public dataset, Messidor-2, is used in experiments. The proposed model has four steps: pre-processing, feature extraction, feature selection, and classification. Firstly, fundus images are pre-processed by resizing images and normalizing pixel values. The inception-v3 model is applied with the transfer learning approach for feature extraction from processed images. Then, classification is made using machine learning algorithms: Extreme Gradient Boosting (XGBoost), Random Forest, Extra Trees, Bagged Decision Trees, Logistic Regression, Support Vector Machines, and Multilayer Perceptron. XGBoost gives maximum accuracy of 91.40%. The best potential features are selected from the extracted features by three metaheuristic algorithms: Particle Swarm Optimization, Simulated Annealing, and Artificial Bee Colony. Selected features are classified with the XGBoost algorithm. The metaheuristics significantly reduced the number of features obtained from each fundus image and increased the classification accuracy. According to the results, the highest accuracy of 93.12% is obtained from the features selected with Particle Swarm Optimization.

When the study results are compared with the existing studies in the literature, it has shown that this study is competitive in terms of accuracy performance and obtained low features. On the other hand, the proposed model has some advantages; it has a few pre-processing steps, training number of parameters are considerable low, and model can be trained with a small amount of data.

This study is one of the first studies showing that better results can be obtained in DR classification by using deep learning and metaheuristic algorithms together. The proposed model can be used to give another idea for ophthalmologists in diagnosing DR.

DİYABETİK RETİNOPATİNİN TANISI İÇİN HİBRİT BİR DERİN ÖĞRENME META-SEZGİSEL MODELİ

ÖZET

Diyabet, pankreasın yeterli insülin üretememesi veya üretilen insülinin etkisinin yetersiz kalması, etkili şekilde kullanılamaması sonucu kan şekerinin artmasıyla sonuçlanan bir hastalıktır.

Uluslararası Diyabet Federasyonu 2021 raporuna göre dünya genelinde yaşları 20 ile 79 arasında değişen yaklaşık 537 milyon yetişkin diyabetle yaşamaktadır. Diyabetli sayısının 2030'da 643 milyona ve 2045'te ise 783 milyona çıkacağı tahmin edilmektedir. Diyabetli yetişkinlerin %75'i düşük ve orta gelirli ülkelerde yaşamaktadır. Diyabetle yaşayan yaklaşık her iki yetişkinden biri (240 milyon hasta) teşhisi konulmamış halde yaşamaktadır. 2021 yılında diyabete bağlı yaklaşık 6.7 milyon ölüm gerçekleşti. Bu hastalık en az 966 milyar dolarlık sağlık harcamasına neden oldu. Yaşları 0-19 arasında değişen çocuk ve gençler açısından bakılacak olursa 1.2 milyondan fazla tip 1 diyabetli vardır. Altı canlı doğumdan biri hamilelik sırasında diyabetten etkinemektedir. 541 milyon yetişkin, tip 2 diyabetli olma riski altındadır.

Diyabetik retinopati (DR), diyabetli kişilerde görme kaybı ve körlüğe neden olabilen bir göz rahatsızlığıdır. Günümüzde körlüğe neden olan hastalıkların başında gelmektedir. Herhangi bir diyabet türü olan herkes DR olabilir. Oftalmolojide, tip 2 diyabet beş yıldan fazla tedavi edilmediğinde, DR yol açabilir. Tip 2 diyabet, vücudun insülinin etkisiz kullanımından kaynaklanır. Diyabetli kişilerin %95'inden fazlası tip 2 diyabetlidir. Bu tip diyabet, büyük ölçüde aşırı vücut ağırlığının ve fiziksel hareketsizliğin sonucudur. Tip 2 diyabet retina görüntülerinde bulunan semptomlarla teşhis edilebilir.

Diyabete bağlı yüksek kan şekeri DR'ye yol açmaktadır. Zamanla kanda çok fazla şeker bulunması retinaya zarar verebilir. Diyabet, vücuttaki tüm damarlara zarar verir. Bu hastalığın göze verdiği zarar, şekerin retinaya giden kılcal damarları tıkayarak sıvı sızdırmasına veya daha ileri aşamada kanamasına neden olmasıyla başlar. Göz, tıkanan damarları telefi etmek için yeni damarlar oluşturur fakat yeni oluşan bu damarlar genellikle iyi çalışmaz, kolayca kanayabilir veya sızdırabilir.

DR'nin erken evrelerinde genellikle herhangi bir semptom görülmez. İlk rahatsızlıklar olarak okuma veya uzaktaki nesneleri görme güçlüğü gibi değişiklikler olabilir. Hastalığın ileriki aşamalarında, retinadaki kan damarları vitreusa kanamaya başlar. Bu durumda hasta çeşitli noktalar, lekeler veya çizgiler görmeye başlayabilir. Bazen bu görüntüler kendiliğinden geçse de tedaviye erken başlanılması önemlidir. Tedaviye başlanmazsa kanama tekrarlayıp, daha kötü olabilir. DR diğer ciddi göz rahatsızlıklarına yol açabilir. Örneğin zamanla, diyabetli yaklaşık 15 kişiden birinde diyabetik makula ödemi gelişir. DR, retinada anormal kan damarlarının oluşmasına yol açabilir ve sıvının gözden dışarı çıkmasını engelleyebilir. Bu bir tür glokoma neden olur.

Diyabet hastalarının yılda en az bir kere kapsamlı bir göz muayenesi yaptırması önemlidir. Diyabetin takibi; fiziksel olarak aktif kalınması, sağlıklı beslenme, ilaçların düzenli kullanılması gibi etkenler göze verilen hasarı durdurabilir, görme kaybının önlenmesine veya geciktirmeye yardımcı olabilir. DR riski diyebet uzadıkça artar. Gebelik, kontrolsüz diyabet, sigara bağımlılığı, hipertansiyon, yüksek kolesterol gibi risk faktörleri DR gelişimini arttırmaktadır. Özellikle diyabetli hamileler mümkün olan en kısa sürede detaylı bir göz muayenesi yaptırmalıdır.

DR göz muayenesinde göz bebeğinin büyütülerek kontrol edilmesiyle tespit edilmesinin yanında görüntü işleme teknikleri yardımıyla da teşhis edilmektedir. DR ve retinada görülen başka hastalıkların tespit edilmesinde fundus floresan anjiografi yöntemiyle elde edilen fundus görüntülerin kullanılması oldukça yaygındır. Günümüzde hasta sayısındaki artışa ve görüntüleme teknolojilerindeki gelişmelere paralel olarak çeşitli yöntemlerle medikal görüntülerden hastalık tespitinin yapıldığı birçok çalışma yapılmaktadır.

Derin öğrenme, uygulama alanı son yıllarda katlanarak artan yöntemlerden biridir. Makine öğrenimi, yapay zekanın bir alt alanıdır. Derin öğrenme de, makine öğreniminin bir alt alanıdır ve yapay sinir ağları, derin öğrenme algoritmalarının temelini oluşturur. Derin öğrenme çok katmanlı bir sinir ağı mimarisi kullanarak girdi bilgilerini birden çok soyutlama düzeyine dönüştürerek, verilerin temsillerini otomatik öğrenen bir yöntemdir. Ağdaki katmanlar problemin çözümüne yönelik öznitelikleri öğrenirler. Öğrenme sırasında hangi bilginin kulllanılacağına yöntem kendisi karar verir.

Derin öğrenme yöntemlerinin çok yönlülüğü, yüksek performansı, yüksek genelleme kapasitesi ve multidisipliner kullanımı gibi özellikleri dikkat çekmektedir. Büyük miktarda verinin toplanması, grafik işleme birimleri, güçlü bilgisayar altyapıları geliştirilmesi, bulut bilişim gibi teknolojik gelişmeler yeni modellerin oluşturulup, uygulanmasını desteklemektedir.

Bölütleme, sınıflandırma, sezim, tanıma, öngörü, doğal dil işleme, örüntü arama, otonom araçlar, veri arıtma gibi görevlerde yüksek performanslar elde edilmektedir. Görüntü elde etme sistemlerindeki gelişmeler, özellikle tıbbi çalışmalarda yüksek çözünürlükte görüntülerin elde edilmesini sağlamaktadır. Tıbbi görüntüler üzerine derin öğrenme çalışmaları yeni sayılır. Dönüm noktası yaklaşk on yıl öncesine, sinir ağlarının geleneksel bilgisayarlı görü yöntemlerinden daha iyi performans göstermeye başladığı döneme uzanır.

Yüksek çözünürlükte görüntülerin artmasının yanında belirli bir hasta vakası için görüntü sayısı, birkaç iki boyutlu görüntüden 3D görüntüleme ile yüzlerce ve 4D dinamik görüntüleme ile binlerce görüntüye yükselmiştir. Artan görüntü sayısıyla uzmanların işyükü de armaktadır. Ayrıca tıbbi görüntülerin karmaşıklığı ve farklı uzmanlar arasındaki varyasyonlar, radyologların ve doktorların her zaman verimli ve doğru teşhis koymasını zorlaştırır.

Derin öğrenmeye dayalı tıbbi görüntü analizinin, çeşitli teşhis ve tedavi süreçlerinde doğruluk ve verimliliğini artırarak klinisyelere karar desteği sağlama konusunda gelecek vaat etmektedir. Günümüzde bazı tıbbi çalışmalarda uzman radyologların başarı seviyeleri yakalanmıştır. Derin öğrenme modelleri örneğin klinik, radyolojik veya patolojik görüntülerin önceden belirlenmiş bir kategoride sınıflandırmayı öğrenir. Verinin, model eğitilmeden önce kategorilere ayrılması işlemi de uzman bir personel tarafından yapılır.

Görüntülerde örüntü tanıma, görüntü/nesne tanıma veya sınıflandırma çalışmalarında evrişimli sinir ağları (ESA) en yaygın kullanılan derin öğrenme ağlarıdır. Evrişimsel ağlar, biyolojik süreçlerden ilham almıştır, nöronlar arasındaki bağlantı modeli, hayvan görsel korteksinin organizasyonuna benzer. ESA'lar çok katmanlı algılayıcıların farklı bir şekilde düzenlenmiş biçimidir. Evrişim, havuzlama ve tam bağlı katmanlardan oluşur. Yeterince büyük bir eğitim seti ile ESA modelleri geriye yayılım algoritmalarını kullanarak veriden ilgili özellikleri çıkarmayı otomatik olarak gerçekleştirebilir. ESA modeli eğitim yoluyla veriden özellik temsillerini keşfeder ve modele girdi olarak manuel olarak tasarlanmış özellikler gerektirmez. Öznitelik çıkarmada ön bilgiden ve insan müdahalesinden bağımsızlık olması ESA modellerinin en önemli avantajlarındandır.

ESA'lar tıbbi görüntü analizinin birçok alanında yaygın olarak kullanılmaktadır. Röntgen, manyetik rezonans görüntüleme, bilgisayarlı tomografi gibi teknolojilerden elde edilen görüntüleri kullanarak bir hastalığın varlığı veya yokluğu, hastalığın türü, sınıflandırması çalışmalarında kullanılmaktadır. Örneğin alzheimer hastalığının sınıflandırması, beyin tümörü sınıflandırması, akciğer nodüllerinin sınıflandırılması, meme kanseri teşhisi, karın bölgesindeki organlarda görülen hastalıkların sınıflandırması, Covid-19 teşhisi, elektroensefalografi sınıflandırması, göz hastalıklarının sınıflandırması gibi.

Bu çalışmada DR'nin otomatik tanısı için bir hibrit model önerildi. Bu modelde derin öğrenme yöntemi olan ESA, sezgi üstü algoritmalar ve makine öğrenmesi algoritmaları ile Messidor-2 açık kaynak fundus görüntüleri kullanılarak DR hastalığının ikili sınıflandırılması yapılmıştır. Önerilen modelin dört adımı vardır. Öncelikle fundus görüntülerinin ön işlemesi yapılarak, görüntüler yeniden boyutlandırılıp, piksel değerleri normalize edilmiştir. Bir derin ESA modeli olan Inception-v3 ile transfer öğrenme yöntemi kullanılarak işlenmiş görüntülerden özellik çıkarıldı. Ardından makine öğrenmesi algoritmaları olan Aşırı Gradyan Artırma, Rastgele Orman, Ekstra Ağaçlar, Torbalama Karar Ağaçları, Lojistik Regresyon, Destek Vektör Makineleri ve Çok Katmanlı Algılayıcı kullanılarak sınıflandırma yapıldı. En yüksek doğruluk performansı %91.40 ile Aşırı Gradyan Artırma algoritmasından elde edildi. Sezgi üstü algoritmalar olan Parçacık Sürü Optimizasyonu, Yapay Arı Kolonisi ve Benzetimli Tavlama algoritmaları kullanılarak, daha önce çıkarılan özelliklerden, en iyi potansiyel özellikler seçildi. Seçilen özellikler Aşırı Gradyan Artırma algoritması ile sınıflandırıldı. Sezgi üstü algoritmalar ile her bir fundus görüntüsünden elde edilen özellik sayısı büyük oranda düşürülüp, sınıflandırma doğruluğu arttırıldı. Sınıflama sonuçlarına göre en yüksek doğruluk, %93.12, Parçacık Sürü Optimizasyonu ile seçilen özelliklerden elde edildi.

Analiz sonuçları literatürdeki mevcut çalışmalarla karşılaştırıldığında, bu çalışmanın doğruluk performansı ve görüntülerden çıkarılan özellik sayıları açısından rekabetçi olduğu gösterilmiştir. Öte yandan önerilen modelin bazı avantajları vardır; görüntülere oldukça az sayıda ön işleme yapılmaktadır, eğitilen parametlerin sayısı azdır ve model az miktarda veri ile başarılı sonuçlar vermektedir.

Bu çalışma, derin öğrenme ve metasezgisel algoritmaların birlikte kullanılmasıyla DR sınıflandırmasında daha iyi sonuçlar alınabileceğini gösteren ilk çalışmalardan biridir. Önerilen model, göz doktorlarına DR teşhisinde destek vermek için kullanılabilir.

1. INTRODUCTION

Diabetes or diabetes mellitus appears when a human body generates high blood glucose levels. The body loses its ability to manage the produced insulin level.

Diabetes causes much damage to patients. There are different types of diabetes. Diabetic retinopathy (DR) is an eye disease caused by diabetes. Specifically, Type 2 is diagnosed with some symptoms in the retina, such as hemorrhages, microaneurysms, exudates, cotton wool spots, etc. (Nagpal et al., 2021).

During the DR, blood vessels that feed the retina tissue are damaged. DR can lead to vision loss or blindness if not diagnosed and treated timely (Bodapati et al., 2021; Farooq et al., 2022). To put it in more detail, if Type 2 diabetes is not treated for more than five years, it causes DR. Reducing some risk factors, particularly hyperglycemia and hypertension, helps control DR progression (Nagpal et al., 2021). Early recognition and screening regularly can reduce visual loss risk to 57%, besides reducing treatment costs (Lakshminarayanan et al., 2021).

The standard diagnosis process of DR involves manual observation of the retina by ophthalmologists. The experts can diagnose DR by inspecting the retina, fundus images, or optical coherence tomography. The inspecting process is complex, takes time, and is error-prone (Bodapati et al., 2021). For example, detecting DR using fundus images requires intense effort because of small-sized lesions and contrast problems (Orlando et al., 2018).

Resnikoff et al. (2020)'s study shows that the estimated mean ophthalmologist density is about 32 per million. Although the number of ophthalmologists is growing, the proper distribution of experts and the development of extensive care systems are required globally. The low ratio of doctors per patient makes it difficult manually detection of DR.

The International Diabetes Federation estimated diabetic patients to be seven hundred million in 2045. According to Teo et al. (2021), the global occurrence of DR is about 22.2% among diabetic patients. The prevalence of vision-threatening cases because of

DR is about 6.1%. In 2020, the number of adults with DR and vision-threatening cases was estimated at 103.1 million and 28.5 million, respectively. By 2045, the numbers are estimated to be about 160 million and 44.8 million, respectively.

The manual observation process and shortage of experts make a challenging situation. The computer-aided diagnosis tools, a subarea of artificial intelligence, have been used in medical tasks since the 1980s. These tools promise ophthalmologists to diagnose DR. They can save cost, time, and required experts for DR screening and can decrease misdiagnosis possibility. On the other hand, they can help manage the growing number of DR patients and diagnose DR early when few vision-threatening effects are seen. The artificial intelligence-based techniques are classified between deep learning and machine learning-based solutions. These techniques offer remarkable solutions for DR diagnosis, severity classification, progression analysis, abnormalities detection, semantic segmentation, etc. (Lakshminarayanan et al., 2021).

The most applied deep learning models on image datasets are Convolutional Neural Networks (CNNs). CNNs originated from the visual cortex of the brain and have superior performance on several tasks, such as image recognition, natural language understanding, speech recognition, automatic video classification systems, or self-driving cars (Buyya et al., 2016).

Heuristics and metaheuristics are approximation methods. These methods can solve nondeterministic polynomial (NP) challenging problems in an acceptable time and promise reasonable solutions. Heuristic algorithms are more task-dependent than metaheuristic algorithms. Metaheuristic algorithms offer a robust solution that combines two search schemas: exploitation and exploration. The exploitation is responsible for finding new searching regions.

On the other hand, the exploration searches for the best solution areas. Metaheuristics as generic algorithms can be applied to many optimization problems (Abdel-Basset et al., 2018; Aydoğan et al., 2019). The feature selection (variable, attribute, or subset selection) is a kind of NP-challenging problem. Various algorithms were proposed for feature selection tasks to achieve close to optimal solutions (Navot et al., 2005; Tahir et al., 2007). Metaheuristics are applied in feature selection tasks and offer superior performance (Yusta, 2009).

This study proposes a hybrid deep learning-metaheuristic model for automated diagnosis of DR. Messidor-2 dataset is used in analyses. Binary classification is made after feature extraction and feature selection processes. A deep learning model, InceptionV3, is used in feature extraction from fundus images after the images are pre-processed. During the feature extraction process, it is taken advantage of the transfer learning approach. Extracted features are classified with Extreme Gradient Boosting (XGBoost), Random Forest, Extra Trees, Bagged Decision Trees, Logistic Regression, Support Vector Machines (SVMs), and Multilayer Perceptron (MLP) algorithms. Particle Swarm Optimization, Simulated Annealing, and Artificial Bee Colony are applied in the feature selection. The feature selection helps increase the model's accuracy by reducing dimensionality and noise in data. Lastly, selected features from three metaheuristic algorithms are classified with XGBoost, giving maximum accuracy in the previous classification task. The proposed model aims to support ophthalmologists in the DR classification task.

Section 2 explains neural networks. Section 3 explains some machine learning concepts regarding this study. Section 4 gives an overview of the related studies in the literature. Section 5 presents the details of methodology. Experimental analysis and results are given in Section 6 and conclusion is given in the last section.

1.1 Diabetic Retinopathy

There are ophthalmic complications of diabetes. One of the most common complications is diabetic retinopathy (DR). The reported prevalence of DR in people with diabetes is about 40%. DR is more common in type-1 diabetes than in type-2. DR occurs up to 90% after thirty years for type-1 diabetics. Diabetes duration is an essential factor in the prevalence of DR. In diabetes patients under 30 years old, DR occurs after ten years is 50%. Regular checkups can help delay or prevent the DR occurrence or progression. Blood glucose should often be controlled. Especially patients with tip-1 benefit from this control more than type-2. Pregnants, renal patients, and high blood pressure patients should pay more attention. Obesity, smoking, anemia, and cataract surgery are other risk factors (Bowling, 2016).

DR is a kind of microangiopathy where tiny blood vessels are especially vulnerable to injury from high levels of glucose. These glucose levels cause retinal capillary damage. On the other hand, immediate hyperglycaemic activities on retinal cells are also

possible reasons (Bowling, 2016). DR emerges at a mild level without obvious visual symptoms. However, progression to severe levels can lead to vision loss. DR is diagnosed by observing the retinal fundus directly or with the help of images obtained from imaging technologies such as optical coherence tomography or fundus photography (Lakshminarayanan et al., 2021).

The Early Treatment Diabetic Retinopathy Study (ETDRS) presents a widely adopted classification for DR. According to ETDRS; two main categories are non-proliferative DR (NPDR) and proliferative DR (PDR). NPDR category includes no DR, very mild, mild, moderate, severe, and very severe NPDR. On the contrary, PDR has mild-moderate, high-risk PDR, and advanced diabetic disease (Bowling, 2016).

There are other categories in widespread use in literature, such as International Clinical Diabetic Retinopathy (ICDR) scale is widely used in clinical and computer-aided research. The ICDR classifies DR in five severity. Figure 1.1 illustrates retinal fundus images for DR stages.

- No DR category has no abnormalities.
- Mild NPDR is the first stage of DR, and microaneurysms are seen in one of four quadrants.
- Moderate NPDR has microaneurysms, exudates, and hemorrhage signs in fundus images. Blood leakage from clogged retinal vessels occurs, and some changes in venules of the retina cause venous beadings.
- In severe NPDR, many of the retinal blood vessels are clogged. More than twenty intraretinal hemorrhages are seen in each quadrant of the fundus, or intraretinal microvascular abnormalities are seen in at least one quadrant. Definite venous beading can occur in at least two quadrants.
- In PDR, new blood vessels, known as neovascularization, are formed. These vessels are usually fragile, so they have a high risk of fluid leakage and the growth of fibrous tissue (Lakshminarayanan et al., 2021; Nagpal et al., 2021).

Figure 1.1 : DR stages (Lakshminarayanan et al., 2021; Nagpal et al., 2021).

DR can be graded by looking at some morphological changes in the retina. DR symptoms are microaneurysms, soft exudates or cotton wool spots, hard exudates, hemorrhages, and the formation of new blood vessels. Moreover, changes occur in the optic disk, macula, and optic nerve head. (Nagpal et al., 2021). Figure 1.2 illustrates a normal retina, and Figure 1.3 shows DR symptoms.

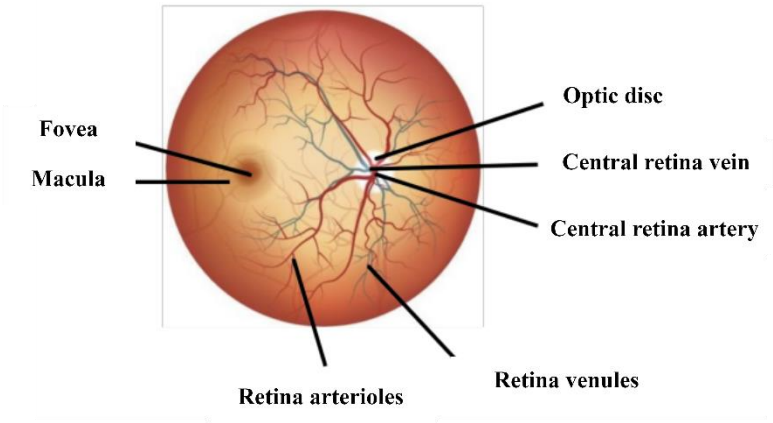


Figure 1.2 : The normal retina (Nneji et al., 2022).

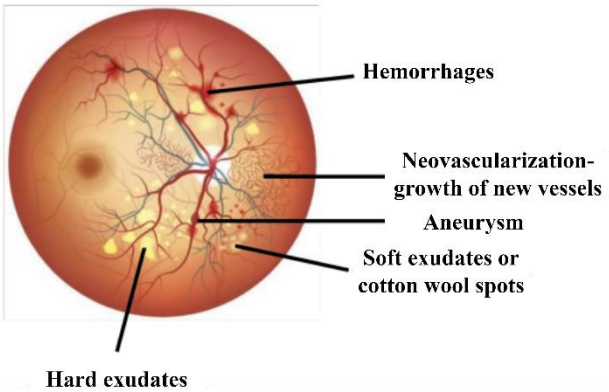


Figure 1.3 : Diabetic retinopathy symptoms (Nneji et al., 2022).

DR severity can be detected either manually by ophthalmologists or automatically by computer-aided diagnosis (CAD). The manual inspection is time-consuming, costly, and necessitates experienced clinicians. Even if all resources, there is a risk of misdiagnosis. The manual process is also challenging in terms of the increasing number of patients with DR; on the contrary, the shortage of ophthalmologists and clinical facilities. CAD techniques, under the scope of artificial intelligence, have become a frequently applied area for ophthalmology. These technologies promise many solutions to fight against DR, such as decreasing misdiagnosis rates, saving time, cost, and other resources, overcoming the increasing number of patients, etc. Artificial intelligence-based techniques are popular in medical research with deep learning and machine learning-based solutions (Lakshminarayanan et al., 2021).

1.2 An Artificial Neuron

An artificial neuron or a perceptron applies a nonlinear mapping from RN to RK , where N is the signal numbers (input) to the neuron, and K is the dimension of the target space. RK is mostly [-1, 1] or [0, 1], based on activation function. The mapping can be defined in equation 1.1:

$$f: R^N \rightarrow R^K \tag{1.1}$$

Figure 1.4 shows an artificial neuron. Received input signals are represented by N, which is a vector. Inputs can be received from other neurons or input data (environment) (1.2).

$$x = x_1, x_2, \dots, x_n \tag{1.2}$$

There is a weight, wi, for each input signal, xi. The corresponding weights strengthen (fire) or terminate the input signals. The neuron calculates the net input signal. The net input signal is the weighted sum of input signals, presented in equation 1.3:

$$\text{Net} = \sum_{i=1}^{N} x_i w_i \tag{1.3}$$

Then, a bias term (b) is added to the net input signal. The bias value also affects the strength of the output signal. An activation function, a(.), is applied to the obtain output signal. A nonlinear activation function transforms the weighted sum. The operation is given in equation 1.4:

$$\text{output} = \text{a}\left(\left(\sum_{i=1}^{N} x_i \times w_i\right) + b\right) \tag{1.4}$$

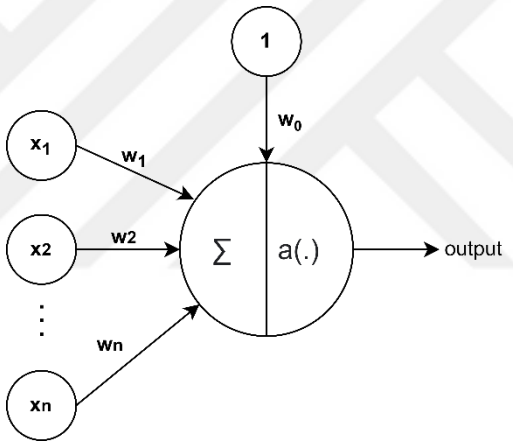


Figure 1.4 : A neuron.

There are various activation functions. In general, activation functions give mappings, which increase monotonically (except for linear function). The CNN section will explain details of frequently used activation functions such as sigmoid, hyperbolic tangent, or Rectified linear unit (ReLU). An artificial neuron learns bias (b) and weights (w_i) from the data. Bias and weight are adjusted until a particular criterion, or various criteria are satisfied.

Artificial neuron learning has three main learning types: supervised, unsupervised, and reinforcement. Simply put, when a dataset includes label (desired output) information corresponding to each sample, this dataset can be used in supervised learning, where the aim is to minimize the error between target output and real output. In unsupervised learning, features or patterns are identified from the dataset with unsorted information.

Samples are neither labeled nor classified. Unsupervised learning concentrates on clustering mainly. Similarities and differences between samples are used. Reinforcement learning is based on a system of rewards and punishments learned from trial and error. The maximum reward is searched (Engelbrecht, 2007).

2. NEURAL NETWORKS

An artificial neural network (ANN) or neural network (NN) approximates a mapping function between inputs and outputs. As the name suggests, it is inspired by a simplification of human brain structure, where the information is propagated through receiving stimuli and transmitting responses using interconnected neurons.

The human brain has an enormous network structure of neurons, which are interconnected. Neurons communicate with each other. A simplified neuron structure is shown in Figure 2.1. A neuron has a soma or cell body, a nucleus, axons, dendrites, axon terminals, and myelin sheaths if explained in simple terms. The nucleus keeps the genetic information of the cell. Axons and dendrites enable the communication of neurons. Electricity is used during communication. Myelin isolates axons and helps electricity transmit through the neuron efficiently. The dendrites and axons are input and output channels of the neuron, respectively. Axons transmit the signal to the dendrites of nearby connected neurons. Inputs can be sent to other connected neurons or have no signal output. The transmission state of the electric pulse depends on some factors.

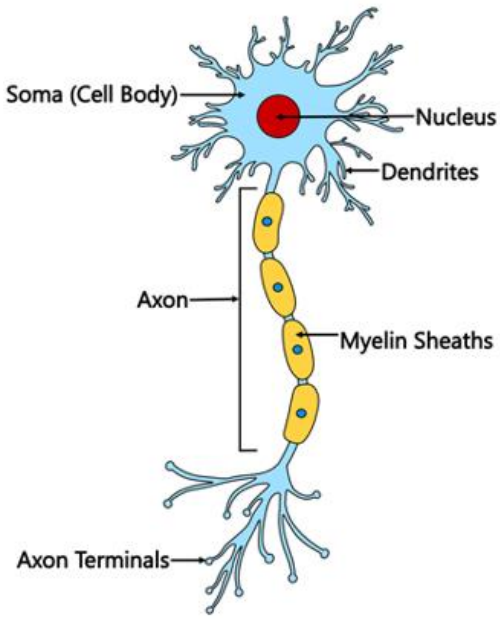


Figure 2.1 : A simplified neuron structure in the human brain.

The typical features of a NN with neurons of the human brain are: it can have a massive number of interconnected neurons, and information is propagated through the network by inputs received from nearby neurons and mapped to outputs. These outputs are sent to be the input of the next layer of neurons. A typical NN structure is given in Figure 2.2. According to Figure 2.2, a NN consists of layers of neurons. The NN has an input, hidden, and output layer. Circular nodes represent artificial neurons. Arrows represent connections and information flow. Each connection between neurons (each arrow) is linked to weight. The weights act on data flowing (URL-1).

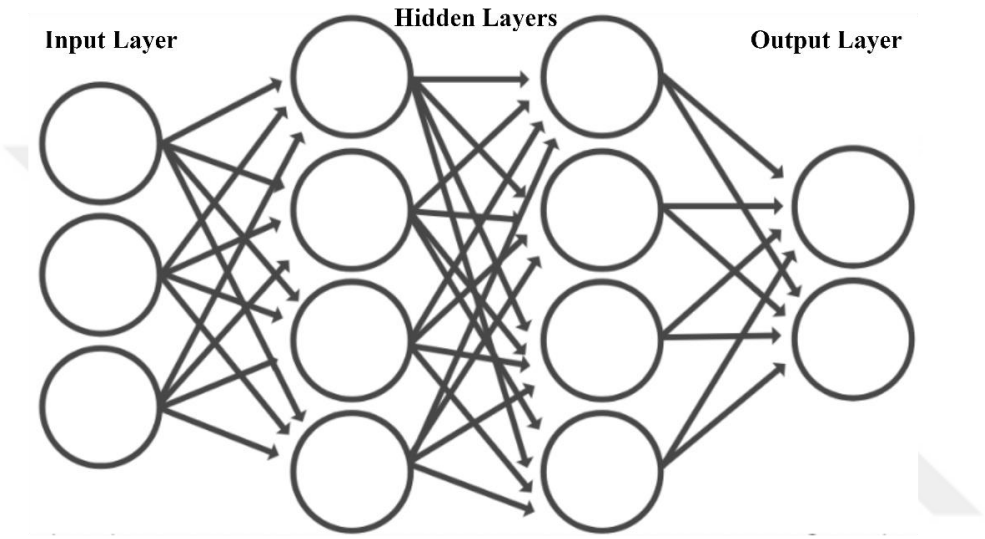


Figure 2.2 : A typical neural network structure.

2.1 Training a Neural Network

Neural network (NN) training searches for weights that enable finding the patterns from data. During the process, gradient descent and backpropagation algorithms are employed. The model gives predictions with given weights. Then these predictions are used in error calculation.

When the network is traversed from the input to the output direction, each time error (difference between expected truth and produced output) is calculated using a loss function. Weights are updated by the backpropagation algorithm, which calculates the gradient of this error to weight changes. Partial derivatives and chain rule are used. Gradient descent algorithm searches to update weights so that the following assessment decreases the error. The gradient of error is navigated down (URL-1).

2.1.1 Gradient descent

The gradient descent is an optimization algorithm to find a differentiable function's local minimum. A gradient descent example for one training parameter is presented in Figure 2.3. Suppose that a loss function is given with one parameter, w. A random value of w is selected initially, and the value of the loss function (L) is calculated. It will probably be high. At the same chosen point, the derivative is calculated, which indicates whether the function has a negative or positive slope. In other words, it is decreasing or increasing. So initial value can be reduced by a derivative fraction to get a new w value that produces a lower loss. This process is iterative. It is repeated until the possible lowest value for loss function is found. When the problem is multivariate, the derivative will be exchanged for a partial derivative concerning each parameter (Anaya-Isaza et al., 2021).

The gradient of loss function gives direction where function has the steepest increase rate, and all parameters are updated ingredient's negative direction based on a learning rate, α. In applications, for example, memory limit requires computing the gradients of loss function in terms of parameters by using mini-batch, a training dataset subset, and applied to updates of the parameter. This technique is called mini-batch gradient descent or stochastic gradient descent (Yamashita et al., 2018).

A single update formula is presented in equation 2.1.

$$w := w - \frac{\partial L}{\partial w}\,\alpha \tag{2.1}$$

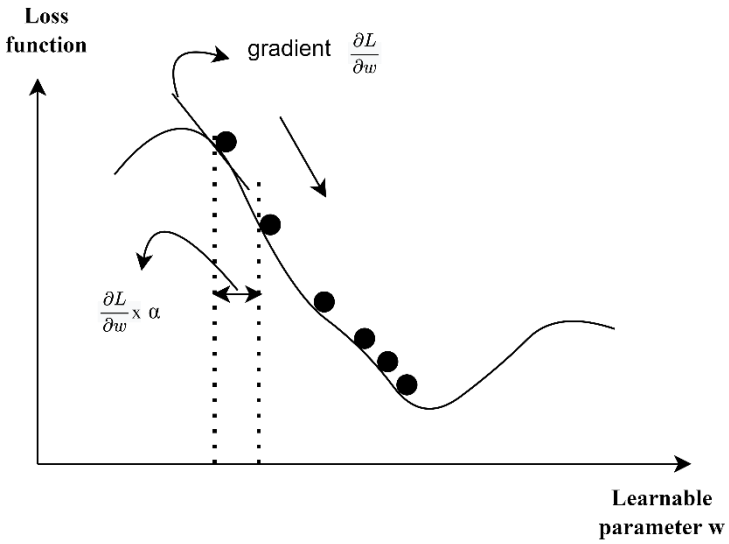


Figure 2.3 : Gradient descent for one parameter.

The activation function should be differentiable and continuous over the real numbers because derivatives of activation functions are calculated during the training. The error gradient is propagated back through the many hidden layers in training a deep NN model. This process can reason the error signal to quickly decrease to zero, in particular when the maximum value of the derivative function is small to start with. This problem is called a vanishing gradient. ReLU function is applied widely in deep learning tasks to handle this problem (URL-1).

A weight update example using two neurons is shown in Figure 2.4.

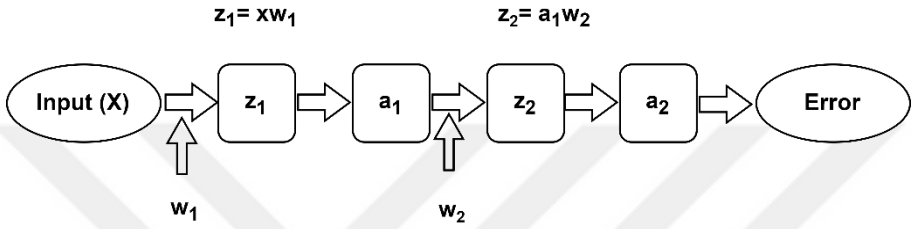


Figure 2.4 : Operations in two neurons.

X is input, a_1 and a_2 are activation functions. z_1 and z_2 are inputs of activation functions. w_1 and w_2 are weights. When the chain rule is applied, firstly overall error of the network is connected to the w_2 as follows (2.2):

$$\frac{\partial_{error}}{\partial_{w_2}} = \frac{\partial_{error}}{\partial_{a_2}} \frac{\partial_{a_2}}{\partial_{z_2}} \frac{\partial_{z_2}}{\partial_{w_2}} \tag{2.2}$$

Secondly, the overall error is connected to the weight, w_1 (2.3).

$$\frac{\partial_{error}}{\partial_{w_1}} = \frac{\partial_{error}}{\partial_{a_2}} \frac{\partial_{a_2}}{\partial_{z_2}} \frac{\partial_{z_2}}{\partial_{a_1}} \frac{\partial_{a_1}}{\partial_{z_1}} \frac{\partial_{z_1}}{\partial_{w_1}} \tag{2.3}$$

After the gradient of the network error is computed as regards each weight, the gradient descent algorithm updates weights for the next forward propagation (at time t+1). For w_1, the weight update formula applying gradient descent is given as follows (2.4) (URL-1), and δ1 is given in equation (2.5):

$$w_1^{t+1} = w_1^t + \left(\eta * \delta_1 * \frac{\partial_{z_1}}{\partial_{w_1}} \right) \tag{2.4}$$

$$\delta_1 = \frac{\partial_{error}}{\partial_{a_2}} \frac{\partial_{a_2}}{\partial_{z_2}} \frac{\partial_{z_2}}{\partial_{a_1}} \frac{\partial_{a_1}}{\partial_{z_1}} \tag{2.5}$$

The following section explains the training of a deep NN in more detail.

2.1.2 Deep layered neural network

A deep NN is comprised of computations carried out by sequential layers. The calculation of an N-hidden layered network is given in equation 2.6, where $f^{(n)}(x)$ is an activation function and provides the output of the related hidden layer; pre-activation function $z^{(n)}(x)$ (2.7) is a linear operation of weight $w^{(n)}$ matrix and bias $b^{(n)}$ vector. g is the final output function (Witten et al., 2016).

$$f(x) = g\left[z^{(N+1)}(f^{(N)}(\dots(f^{(2)}(z^{(2)}(f^{(1)}(z^{1}(x))))))\right] \quad (2.6)$$

$$z^{(n)}(x) = w^{(n)}x + b^{(n)} \quad (2.7)$$

An activation function ($f^{(n)}(x)$) run on a pre-activation vector in an elementwise manner. Figure 2.5 presents some activation functions and their graphs. The activation functions commonly ensure a nonlinear relation between the neuron output and weighted input. These functions are generated to separate the values that are generally very close, which allows forming of more space (differentiable) between values. An activation function is a hyperparameter for models (Anaya-Isaza et al., 2021).

ReLU function has been the most preferred in recent deep learning studies because that is helpful in vanishing gradient problem, as mentioned earlier. Moreover, piecewise linear (pw_linear) functions have also been popular (Witten et al., 2016).

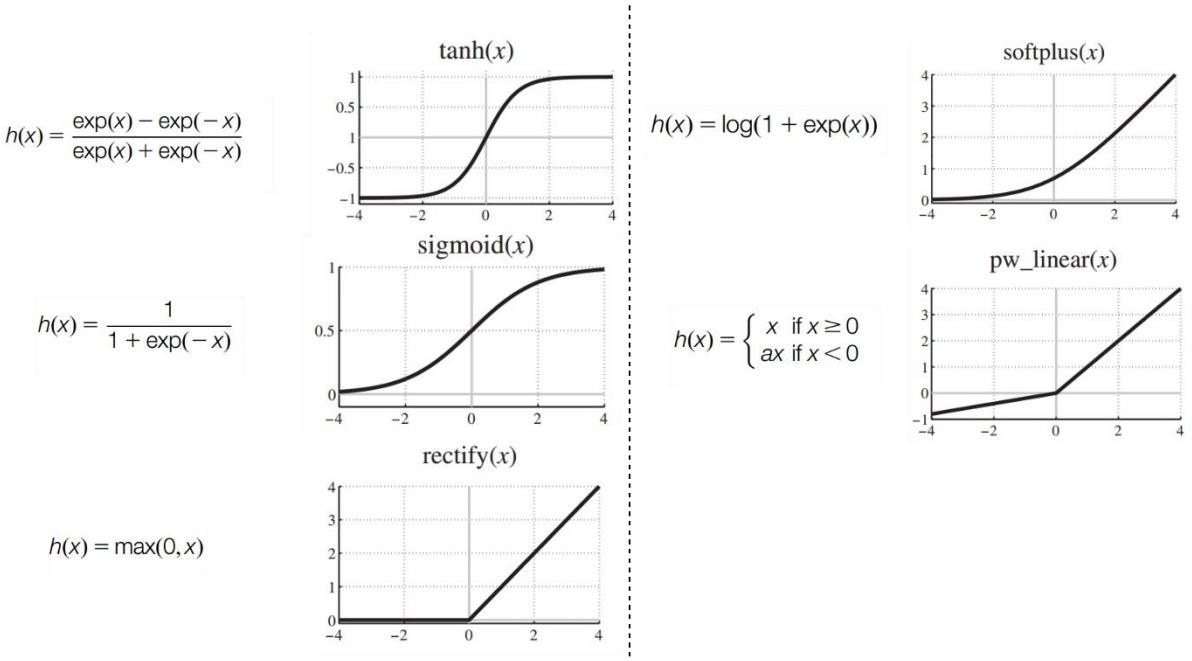


Figure 2.5 : Some activation functions and their graphs (Witten et al., 2016).

2.1.3 Backpropagation process

A typical forward propagation process in a deep network is presented in Figure 2.6. Output is obtained sequentially using weights, biases, pre-activation, and activation functions and loss is calculated. The following backpropagation process is explained in Witten et al. (2016).

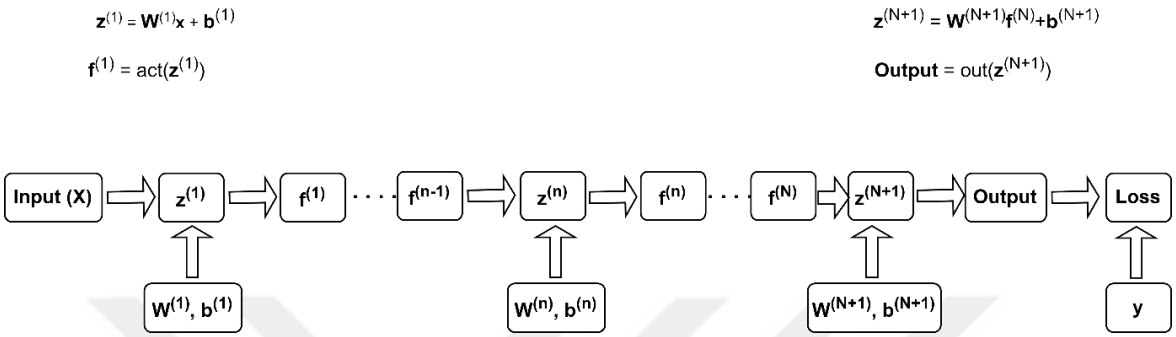


Figure 2.6 : Forward propagation process in a deep network.

Backpropagation applies the chain rule in calculus. For a given single-layer network that has a softmax output (a logistic regression model), the loss can be expressed as $L(g(x; \Phi), y)$. Define $g = [g_1(z), g_2(z), \ldots, g_M(z)]$, and $z(x; \Phi) = [z_1(x; \Phi_1), z_2(x; \Phi_2), \ldots, z_M(x; \Phi_M)]$ with $z_m(x; \Phi_m) = \Phi_m^T x$ where Φ_m is a column vector and contains m^{th} row of Φ parameter matrix. A softmax loss for $g(z(x))$ is presented in equation (2.8).

$$\text{Loss} = -\sum_{m=1}^{M} y_m \log g_m(\mathbf{x}), \qquad g_m(\mathbf{x}) = \frac{\exp(z_m(\mathbf{x}))}{\sum_{r=1}^{M} \exp(z_r(\mathbf{x}))} \tag{2.8}$$

$z_m(x; w_m, b) = w_m^T x + b$ where x contains a 1 in the last; therefore, b (bias) parameter is the last unit of each parameter vector Φ_m. The chain rule can be applied in vector form in equation 2.9, presenting the partial derivative relative to any specified parameter vector Φ_m.

$$\frac{\partial L}{\partial \boldsymbol{\theta}_m} = \frac{\partial \mathbf{z}}{\partial \boldsymbol{\theta}_m} \frac{\partial \mathbf{g}}{\partial \mathbf{z}} \frac{\partial L}{\partial \mathbf{g}} = \frac{\partial \mathbf{z}}{\partial \boldsymbol{\theta}_m} \frac{\partial L}{\partial \mathbf{z}} \tag{2.9}$$

Each component of ∂L relative to z is presented in equation (2.10).

$$\frac{\partial L}{\partial z_j} = \frac{\partial}{\partial z_j} \left[-\sum_{m=1}^{M} y_m \left(z_m - \log \left(\sum_{r=1}^{M} \exp(z_r) \right) \right) \right] \tag{2.10}$$

$$= -\left[y_{m=j} - \frac{\exp(z_{m=j})}{\sum_{r=1}^{M}\exp(z_r)}\right]$$

$$= -[y_j - p(y_j \mid \mathbf{x})] = -[y_j - g_j(\mathbf{x})]$$

The vector form is obtained as follow: $\partial L/\partial z = -[y-g(x)]$, where $\mathcal{E} = [y-g(x)]$ is frequently called as error. Since equation 2.11 implies equation (2.12), where the vector x is found in m^{th} column in the matrix.

$$\frac{\partial z_j}{\partial \boldsymbol{\theta}_m} = \begin{cases} \frac{\partial}{\partial \boldsymbol{\theta}_m}\boldsymbol{\theta}_m^T\mathbf{x} = \mathbf{x}, & j = m \\ 0, & j \neq m \end{cases} \tag{2.11}$$

$$\frac{\partial \mathbf{z}}{\partial \boldsymbol{\theta}_m} = \mathbf{F}_m = \begin{bmatrix} 0 & x_1 & 0 \\ \vdots & \vdots & \vdots \\ 0 & x_k & 0 \end{bmatrix} \tag{2.12}$$

Equation 2.13 provides gradient for the vector in m^{th} row in the matrix.

$$\frac{\partial L}{\partial \boldsymbol{\theta}_m} = \frac{\partial \mathbf{z}}{\partial \boldsymbol{\theta}_m}\frac{\partial L}{\partial \mathbf{z}} = -\begin{bmatrix} 0 & x_1 & 0 \\ \vdots & \vdots & \vdots \\ 0 & x_k & 0 \end{bmatrix}[\mathbf{y} - \mathbf{g}(\mathbf{x})] = \mathbf{x}(y_m - g_m(x)) \tag{2.13}$$

The gradient of the whole matrix of parameters Φ can be computed as in equation (2.14).

$$\frac{\partial L}{\partial \boldsymbol{\theta}} = -(\mathbf{y} - \mathbf{g}(\mathbf{x}))\mathbf{x}^T = -\mathcal{E}\mathbf{x}^T \tag{2.14}$$

For a deep layered NN, suppose an identical activation function is used for all N hidden layers, and softmax is used in the output layer. The gradient of m^{th} parameter vector of $(N+1)^{th}$ parameter matrix is presented in equation 2.15, where F_m^N is a matrix consisting of activations of the related hidden layer (column m), and $\mathcal{E}^{(N+1)}$ is the error of the output layer.

$$\frac{\partial L}{\partial \boldsymbol{\theta}_m^{(N+1)}} = \frac{\partial \mathbf{z}^{(N+1)}}{\partial \boldsymbol{\theta}_m^{(N+1)}}\frac{\partial L}{\partial \mathbf{z}^{(N+1)}} \; ; \; \frac{\partial L}{\partial \mathbf{z}^{(N+1)}} = -\mathcal{E}^{(N+1)}$$

$$= -\frac{\partial \mathbf{z}^{(N+1)}}{\partial \boldsymbol{\theta}_m^{(N+1)}}\mathcal{E}^{(N+1)} = -\mathbf{F}_m^N\mathcal{E}^{(N+1)} \tag{2.15}$$

The complete parameter updating is reorganized in equation (2.16).

$$\frac{\partial L}{\partial \boldsymbol{\theta}^{(N+1)}} = -\mathcal{E}^{(N+1)}\tilde{\boldsymbol{f}}_{(N)}^T \tag{2.16}$$

The following step is the backpropagation of the error term. Assume that gradient of m^{th} row of N^{th} parameters matrix. Because biases are constant terms, they are not backpropagated. $\mathcal{E}^{(N)}$ is given relative to $\mathcal{E}^{(N+1)}$ in equation (2.17).

$$\frac{\partial \mathrm{L}}{\partial \boldsymbol{\theta}_{\mathrm{m}}^{(\mathrm{N})}} = \frac{\partial \mathbf{z}^{(\mathrm{N})}}{\partial \boldsymbol{\theta}_{\mathrm{m}}^{(\mathrm{N})}} \frac{\partial \mathbf{f}^{(\mathrm{N})}}{\partial \mathbf{z}^{(\mathrm{N})}} \frac{\partial \mathbf{z}^{(\mathbf{N+1})}}{\partial \mathbf{f}^{(\mathbf{N})}} \frac{\partial \mathrm{L}}{\partial \mathbf{z}^{(\mathrm{N+1})}} \tag{2.17}$$

$$= -\frac{\partial \mathbf{z}^{(\mathrm{N})}}{\partial \boldsymbol{\theta}_{\mathrm{m}}^{(\mathrm{N})}} \frac{\partial \mathbf{f}^{(\mathrm{N})}}{\partial \mathbf{z}^{(\mathrm{N})}} \frac{\partial \mathbf{z}^{(\mathrm{N+1})}}{\partial \mathbf{f}^{(\mathrm{N})}} \mathcal{E}^{(\mathrm{N+1})}; \quad \mathcal{E}^{(\mathrm{N})} \equiv \frac{\partial \mathbf{f}^{(\mathrm{N})}}{\partial \mathbf{z}^{(\mathrm{N})}} \frac{\partial \mathbf{z}^{(\mathrm{N+1})}}{\partial \mathbf{f}^{(\mathrm{N})}} \mathcal{E}^{(\mathrm{N+1})}$$

$$= -\frac{\partial \mathbf{z}^{(\mathrm{N})}}{\partial \boldsymbol{\theta}_{\mathrm{m}}^{(\mathrm{N})}} \mathcal{E}^{(\mathrm{N})}$$

Furthermore, for $n \leq N$ the other $\mathcal{E}^{(n)}$s are defined in iterative way with regard to $\mathcal{E}^{(n)}$ as in equation 2.18, where $P^{(n)}=\partial f^{(n)}/\partial z^{(n)}$, $P^{(n)}$ matrix includes partial derivatives of activation function of hidden layer wrt preactivation input and $W^{T(n+1)} = \partial z^{(n+1)}/ \partial f^{(n)}$. $W^{T(n+1)}$ is produced from $z^{(n+1)}(f^{(n)}) = W^{(n+1)}f^{(n)} + b^{(n+1)}$.

$$\mathcal{E}^{(\mathrm{n})} = \frac{\partial \mathbf{f}^{(\mathrm{n})}}{\partial \mathbf{z}^{(\mathrm{n})}} \frac{\partial \mathbf{z}^{(\mathrm{n+1})}}{\partial \mathbf{f}^{(\mathrm{n})}} \mathcal{E}^{(\mathrm{n+1})} = \mathbf{P}^{(\mathrm{n})} \mathbf{W}^{\mathrm{T(n+1)}} \mathcal{E}^{(\mathrm{n+1})} \tag{2.18}$$

The gradients for the m^{th} parameters vector of the n^{th} network layer can be computed by multiplication of matrices as in equation (2.19).

$$\frac{\partial \mathrm{L}}{\partial \boldsymbol{\theta}_{\mathrm{m}}^{(\mathrm{n})}} = -\mathbf{F}_{\mathrm{m}}^{(\mathrm{n-1})} \mathbf{P}^{(\mathrm{n})} \mathbf{W}^{\mathrm{T(n+1)}} \ldots \mathbf{P}^{(\mathrm{N})} \mathbf{W}^{\mathrm{T(N+1)}} \mathcal{E}^{(\mathrm{N+1})} \tag{2.19}$$

Given the presented equations, g(x) definition, a loss function, and several regularizations, deep layered neural networks are optimized by applying gradient descent. The iterative property of $\mathcal{E}^{(n)}$ indicates the propagation of information back from the loss.

The presented equations are compatible with numerical optimization. Matrix-vector multiplication can be used instead of matrix-matrix multiplication as follow in 2.20.

$$\mathcal{E}^{(n)} = P^{(n)}\left(W^{T(n+1)} \mathcal{E}^{(n+1)}\right) \tag{2.20}$$

Additionally, elementwise multiplication can be used instead of matrix-vector multiplication because many activation functions of the hidden layer produce a diagonal form for $P^{(n)}$ as follow in 2.21.

$$\mathcal{E}^{(n)} = p^{(n)}\Theta\left(W^{T(n+1)}\mathcal{E}^{(n+1)}\right) \tag{2.21}$$

In equation 2.21, Θ is elementwise, and $p^{(n)}$ vector is generated by obtaining elements from the diagonal of $P^{(n)}$.

At each level, the complete parameter matrix is updated in equation (2.22). When n=1, $\hat{\mathbf{f}}_{(0)} = \hat{\mathbf{x}}$, input with a 1 appended.

$$\frac{\partial \mathrm{L}}{\partial \boldsymbol{\theta}^{(\mathrm{n})}} = -\mathcal{E}^{(\mathrm{n})}\hat{\mathbf{f}}^{\mathrm{T}}_{(\mathrm{n}-1)} \tag{2.22}$$

Figure 2.7 presents error propagation or backward computation, where the forward process is illustrated with bold arrows.

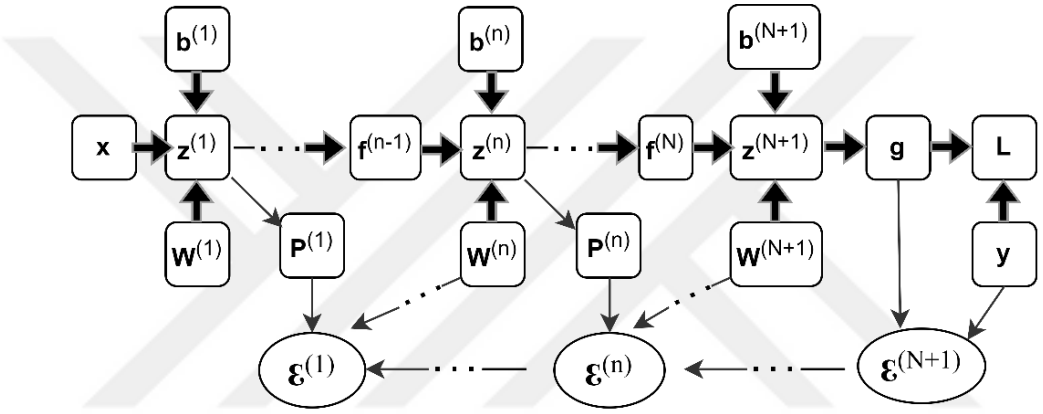
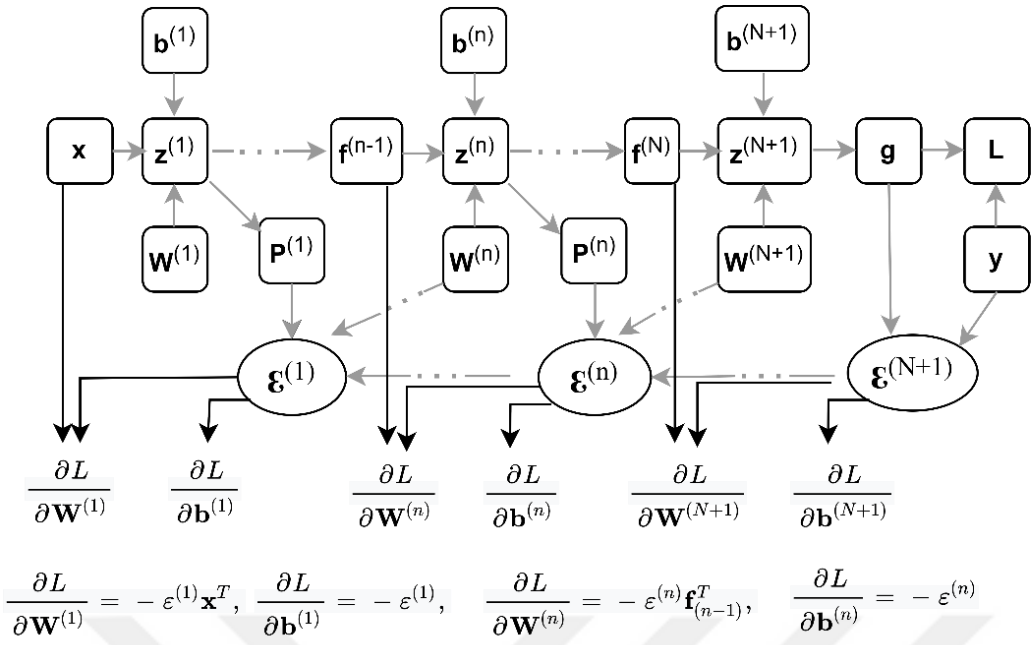


Figure 2.7 : The backpropagation process in a deep neural network.

Figure 2.8 presents the final computations needed for gradient type learning, where updating of parameters tracks forward & backward propagation phases.



$$\frac{\partial L}{\partial \mathbf{W}^{(1)}} = -\varepsilon^{(1)}\mathbf{x}^T, \quad \frac{\partial L}{\partial \mathbf{b}^{(1)}} = -\varepsilon^{(1)}, \quad \frac{\partial L}{\partial \mathbf{W}^{(n)}} = -\varepsilon^{(n)}\mathbf{f}_{(n-1)}^T, \quad \frac{\partial L}{\partial \mathbf{b}^{(n)}} = -\varepsilon^{(n)}$$

Figure 2.8 : Updating of parameters tracks forward & backward propagation phases.

3. MACHINE LEARNING

Machine Learning (ML) is a branch of Artificial Intelligence. In this technique, the model is learned from data such as images, audio, signals, and documents without humans doing it. The model is trained on data to solve a given task. It is a learning process, and the data used in the modeling process is called training data. ML promises to solve problems where analytical models are scarcely present. When laws or equations are not favorable in problem-solving, ML aims to offer a model to achieve this problem using training data. Such as, in a speech recognition task, math equations or physical law do not offer a model.

ML training process is shown in Figure 3.1, where horizontal flow presents the learning process, and vertical flow presents the inference or trained model.

In some cases, input (actual field) and training data can be very distinct. The level of distinctness is a challenge for ML. It is crucial to have unbiased training data that adequately represents field data's characteristics. The generalization of the model is critical in evaluating model performance. The generalization process makes a model's performance steady irrespective of the input or training data (Kim, 2017).

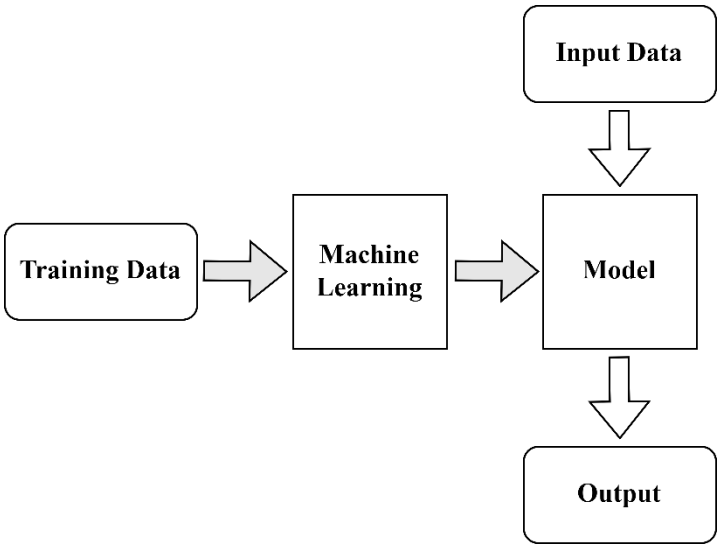


Figure 3.1 : Machine learning process.

Data is often split into training, validation, and test sets in ML or deep learning applications. The training data is used to train a model, in which loss is calculated through forward propagation and parameters are updated through backpropagation. The validation data is used to evaluate the model during training, fine-tune hyper-parameters, and carry out model selection. Lastly, the test set is used at the end of the task to evaluate the final model's performance. Although the model's learnable parameters are not trained directly on the validation set, overfitting can occur on the validation set. Fine-tuned hyperparameters on validation data reason a good performance for model on the same validation data. So, unseen test data is required for the generalizability of the model (Yamashita et al., 2018).

3.1 Overfitting

Generalization can be damageable by overfitting. When every sample of training data fits the model, a low generalization model can be obtained. This situation is an overfitting case (Kim, 2017). When overfitting occurs, the model finds statistical regularities of training data where irrelevant noise is memorized. When a model's performance is good for the training dataset in the comparison validation dataset, it is likely to have overfitted the training dataset (Yamashita et al., 2018). On the other hand, if a model's performance is poor for both validation and training sets, then the underfitting problem is raised. Figure 3.2 shows underfitting and overfitting cases in terms of loss value against the number of iterations.

There is a dilemma in decreasing the model's error rate using training data and increasing the model's generalization (Kim, 2017).

Figure 3.2 : Underfitting and overfitting cases.

Figure 3.3 presents an example of overfitting, where the related model groups the points perfectly for training data.

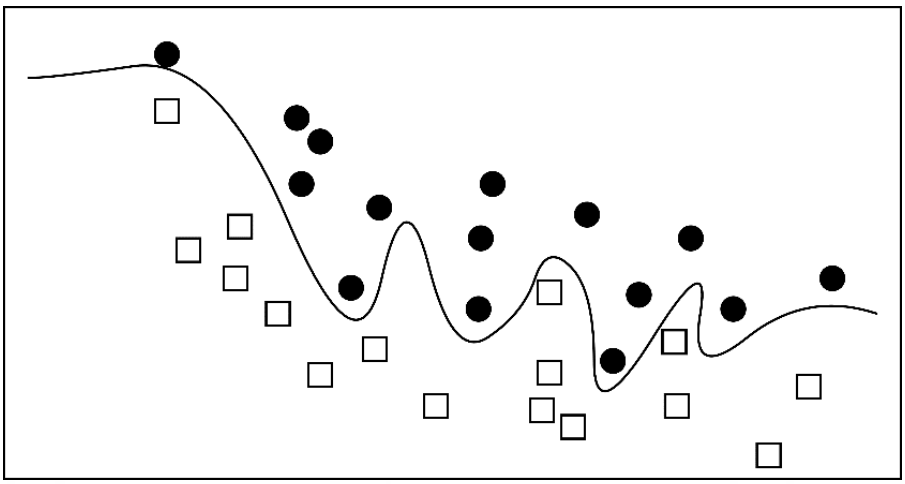


Figure 3.3 : A model that groups the points perfectly.

There are various techniques to minimize overfitting.

The regularization method tries to build a model as simply as possible to prevent overfitting. Like the model curve in Figure 3.3, a complex model tends to overfit. Mathematically, regularization can be made by adding the summation of weights to the cost function.

Some training datasets can have higher dimensions, and models fitted on these datasets cannot be easily visualized. So overfitting cannot be evaluated. In these cases, another method is required to determine the overfitting. It is validation method. Training data is divided into training and validation sets during the validation process. The ratio is generally 4:1 for training and validation sets, respectively. After the data split, the model is trained with training data, and the model's performance is observed on validation data. If the performance is sufficient, training can be terminated. If not, some changes to the model are required (Kim, 2017).

There is also the cross-validation or k-fold cross-validation method, which is a resampling procedure. It is a slightly different version of the validation method. This method's overall process is as follows: Dataset is mixed randomly. The dataset is split into a number of k groups or folds (usually five or ten); these are non-overlapping. For each unique fold, a fold is taken as a test dataset, and the remaining folds are taken as a training dataset. The model is fitted to the training dataset and evaluated on the test

dataset. Lastly, results are summarized with the mean of some model performance scores, such as error rate, accuracy score, etc.

An example of 5-fold cross-validation is presented in Figure 3.4.

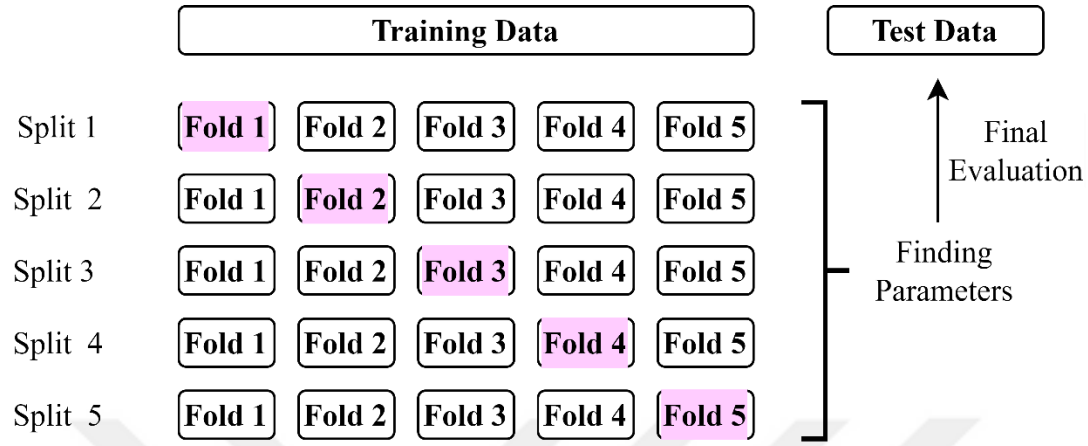


Figure 3.4 : An example of 5-fold cross-validation.

The amount of training data can help the generalization of the model. Reducing the complexity of the model, applying data augmentation methods (modifying training data), dropout (dropping out randomly inputs to a layer such as activations during training), or batch normalization (normalizing input values) can mitigate overfitting (Yamashita et al., 2018).

3.2 Deep Learning

Deep learning (DL) is a type of ML technique. Its theorization goes back to the 1980s. Recently, DL has been very popular, and it has many applications in such areas: driverless cars, medical devices, industrial automation, aerospace & defense, or voice control technologies. DL networks can handle a large amount of data successfully, and the performance of deep networks has bettered classical methods or human-level performance in many fields, particularly in pattern recognition tasks (Albawi and Al-Zawi, 2017).

In a DL model, classification tasks are learned directly from text, image, or sound data. A large number of the dataset with labels are required to train networks with many layers. Additionally, considerable computing power is needed. High-capacity graphics processing units and cloud computing technologies enable the development of deep learning models by speeding up model training. DL models are mainly designed with

neural networks, so DL models are called deep neural networks. While a small neural network has 3-4 layers, DL architectures can have hundreds of layers. DL models are capable of learning features from data directly, without requiring manual processes for feature extraction.

On the other hand, a machine learning process begins with manually extracted features from images. DL performs tasks that have end-to-end learning. Another difference is that DL models scale with data while machine learning models converge when training data size increases (URL-2).

Convolutional Neural Networks (CNNs) are one of the most used DL networks. CNNs are a specialized neural network model. Convolution is a linear mathematical operation between matrixes. This operation gives the network its name (Albawi and Al-Zawi, 2017). The main difference between a CNN model and a feed-forward neural network is that a CNN model assumes inputs as image-like, enabling the encoding of some properties in the model. The following section explains CNNs in more detail.

3.3 Convolutional Neural Network

CNN is a type of DL model and is inspired by the system of an animal visual cortex. CNNs successfully process data that has a pattern, like images. CNN models work with different numbers of dimensional data: one, two, or three. Especially, a CNN model can handle tasks of two-dimensional image datasets. CNNs can extract features from data automatically and learn a spatial hierarchy of features adaptively from low to high levels. That means manual operations do not need to detect these features (Yamashita et al., 2018).

A CNN is comprised of multiple sequences of layers. Typical layers of a CNN are convolution, pooling, and fully connected layers (FCLs). These layers ensure feature extraction, dimension reduction, and prediction or classification, respectively (Teuwen and Moriakov, 2020).

The other layers are on-demand layers, which can make a CNN more successful by decreasing overfitting and increasing generalizability. These secondary layers are batch normalization, dropout, and regularization (Kandel and Castelli, 2020).

Unlike classic neural networks, neurons in the layers of a CNN model are arranged in some dimensions: height, width, channels, and the number of kernels or filters in two

dimensions. In the sequence of layers, every layer transforms outputs or activations of the previous layer through other differentiable functions (Teuwen and Moriakov, 2020).

The most notable feature of a CNN is to decrease the number of parameters in deep learning. This property enables researchers to build deeper models for complex tasks, which are hard to solve for classic networks (Albawi and Al-Zawi, 2017). The optimization process becomes more straightforward, and dependence on data size decreases (Teuwen and Moriakov, 2020).

Operation details of the layers are explained in the following section.

3.3.1 Convolution layers

Convolutional layers are the central part used in CNNs. These layers give the network its name, consisting of nonlinear and linear operations: activation function and convolution operation.

3.3.1.1 Convolution

Convolution is a linear operation. Like usual neural networks, a set of weights is multiplied with an array of input values. The two-dimensional (2D) array of weights is multiplied with the input array called a kernel or a filter. A filter consists of a set of weights. During the training, the network learns filters automatically. That means the CNN model will learn which features or patterns to extract from the data. This ability is the innovation of CNNs.

In a CNN model, multiple filters are learned parallel to extract specific features from a given data, such as an image. A filter's depths (number of channels) should be equal depths of input data. The filter size is smaller than the input size. Element-wise multiplication is done with the filter and same-sized patch of input. Then dot product is summed, and a single value is obtained. Filters behave like a detector of the feature of interest. Using a smaller filter size than the input size enables multiplying the input array by the same set of weights (filter) at different input points. The filter is moved systematically, paying attention to overlapping parts of the filter-sized patch of input and moving directions: from left to right and top to bottom. This systematic application of the same filter through an image enables the filter to detect intended features in the image regardless of location. This property is called translation invariance (URL-3).

In invariance to local translation, the presence of a feature in an image is more important than where the feature is. Such as, in an image, we are trying to detect whether there is a face. In terms of eyes as a feature, we need to detect whether there are two eyes in the face (left and right sides) rather than detect the location of the eyes definitively (Goodfellow et al., 2016).

The same filter application, namely multiplying, is repeated on input by dragging the filter, and this operation results in a feature map. A feature map is a 2D array of output. An example of filter application on a two-dimensional input and forming of a feature map is presented in Figure 3.5, where the stride is 1.

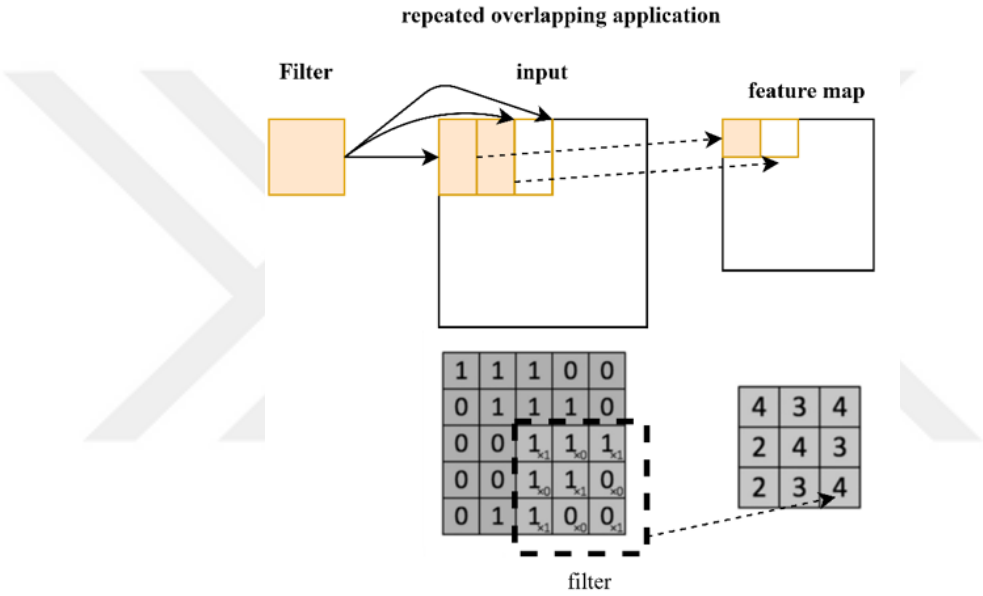


Figure 3.5 : An example of a 2D convolution operation.

A convolution operation has two hyperparameters: the number and size of filters. The mainly applied filter sizes are 3×3, 5×5, or 7×7. Usually, odd numbers are used. The number of a filter is arbitrary. A convolution operation typically produces a feature map that is smaller in width and height than previous feature maps or input tensors. Applying zero padding helps to retain the same in-plane dimensions. Recent CNN applications apply zero padding. Weight sharing is the foremost property of convolution operation. Filters are shared across all image positions, enabling translation invariance. Learning feature patterns in a spatial hierarchy with a pooling operation helps to capture a larger view field and decrease the number of parameters (Yamashita et al., 2018).

After the feature map is generated, values in the feature map are sent to a nonlinear function such as tanh, ReLU, or sigmoid.

Figure 3.6 shows two layers of CNN with different filters and compact representation. The number of produced feature maps depends on the applied number of filters. In the first convolution operation, two filters sized 3×3 are applied, and two feature maps are produced. The number of strides and filters in layers gets the desired value. The stride is related to the number of hops that the filter will move along. A stride is a distance between two consecutive filter positions. It enables downsampling. The depths of filters change according to the number of maps they match.

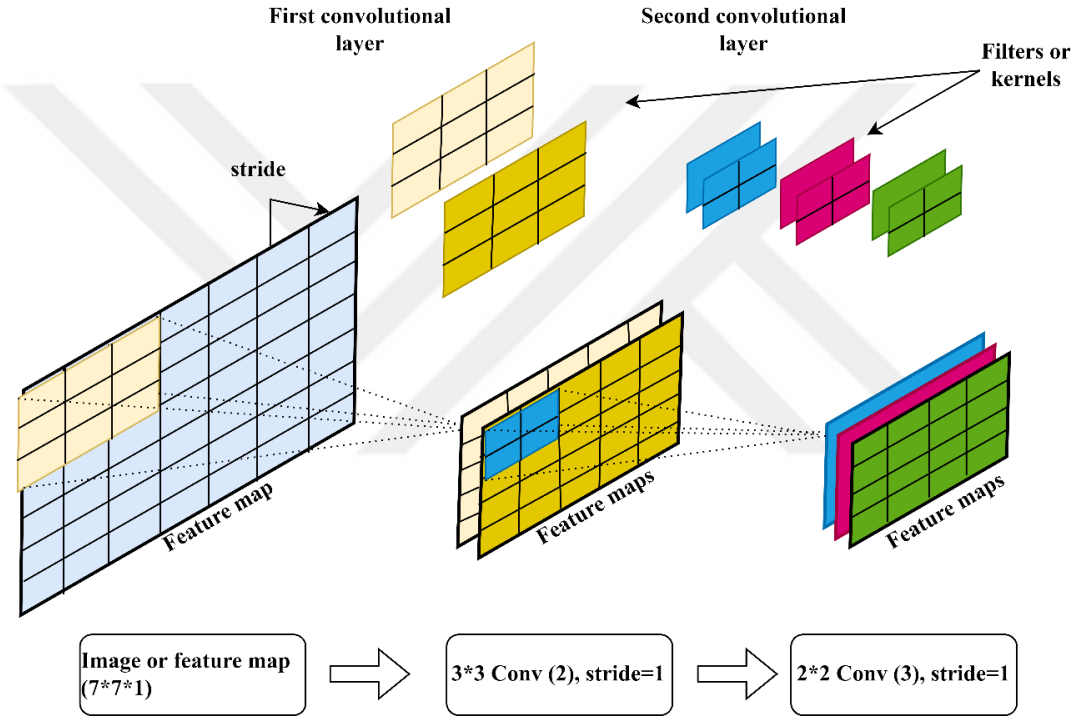


Figure 3.6 : An example of two layers of CNN (Anaya-Isaza et al., 2021).

3.3.1.2 Nonlinear activation function

Input signal or outputs of convolution operation, a real number, is transformed with a nonlinear activation function, and the obtained output is sent to the next layer as an input. An activation function helps to decide which neurons to fire or not fire. A network learns non-linear mappings with activation layers.

There are different types of activation functions used in CNNs in the literature. CNN models' most preferred activation functions are Sigmoid, Softmax, ReLU, hyperbolic tangent (Tanh), and Leaky ReLU. Activation functions are categorized into two groups: saturated and non-saturated. When the output of the activation function takes

values in finite boundaries, it is called a saturated activation function such as sigmoid, Tanh. On the other hand, if the output values are infinite, it is called the non-saturated activation layer. Such as ReLU. Non-saturated activation functions are more advantageous than saturated ones in the vanishing gradient problem encountered in the backpropagation algorithm (Kandel and Castelli, 2020).

The graphs of Tanh, Sigmoid, and ReLU are presented in Figure 2.5. An example of the Leaky ReLU function is presented in Figure 3.7. Softmax is applied to a network's final layer and is widely used in multiclass classification problems. It converts the output of the last layer (a vector of real numbers) into a probability distribution vector.

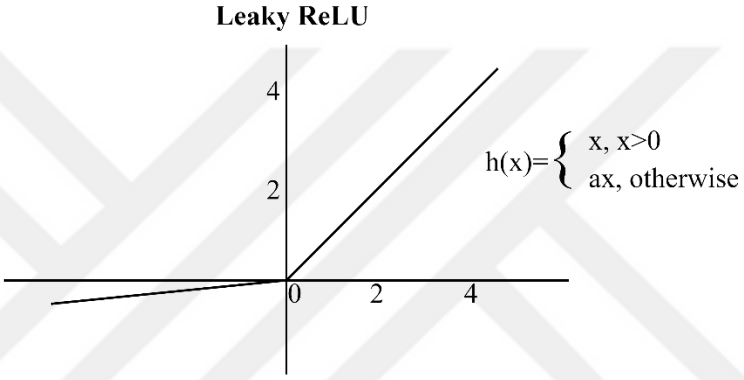


Figure 3.7 : Leaky ReLU activation function.

3.3.1.3 Convolution Process

Mathematically, the convolution process in a CNN model can be expressed in equation 3.1, in which * presents the convolution operation between n^{th} map X_n and filter F_{n1} at the equivalent map depth for N feature maps. f and b1 are activation function and bias, respectively.

$$C_1 = f\left(\sum_{n=1}^{N} X_n * F_{n1} + b_1\right) \tag{3.1}$$

In general, each feature map produced by k^{th} kernel in l^{th} network layer is used by the following equation (3.2). Like neural networks, an input to a layer is output from a previous layer (Anaya-Isaza et al., 2021).

$$C_k^{(l)} = f^{(l)}\left(\sum_{n=1}^{N^{(l-1)}} C_n^{(l-1)} * F_{nk}^{(l)} + b_k^{(l)}\right) \tag{3.2}$$

3.3.2 Pooling layers

A pooling layer is generally added to the model after the convolutional layer. The pooling layer applies pooling operation on each feature map and produces pooled feature maps. This layer decreases the spatial size of a convolution layer's representations or output feature map. It is a downsampling operation. The number of parameters as well as training time is decreased considerably. The pooling operation removes noise and extracts notable features from feature maps. These properties strengthen the spatial invariance of CNN networks. Pooling layers use padding, stride, and filter size as parameters (Kandel and Castelli, 2020). On the other hand, there are no learnable parameters in pooling layers (Yamashita et al., 2018).

2×2 filters with stride of two are the most preferred choice for pooling operation. When such a pooling layer is applied to a feature map, the size of the feature map will be halved, and the depth of the feature map remains the same. Such as, when a pooling layer is applied on a feature map size of 8 × 8, the output will be a 4×4 pooled feature map. There are two common types of pooling operations: average pooling and maximum pooling. Maximum pooling produces the output by finding the maximum value in each patch (filter); that is, the most activated presence of a feature is summarized. Average pooling produces the output by calculating the average value for each patch; in other words, the average presence of a feature is summarized.

An example for both pooling operations is given in Figure 3.8.

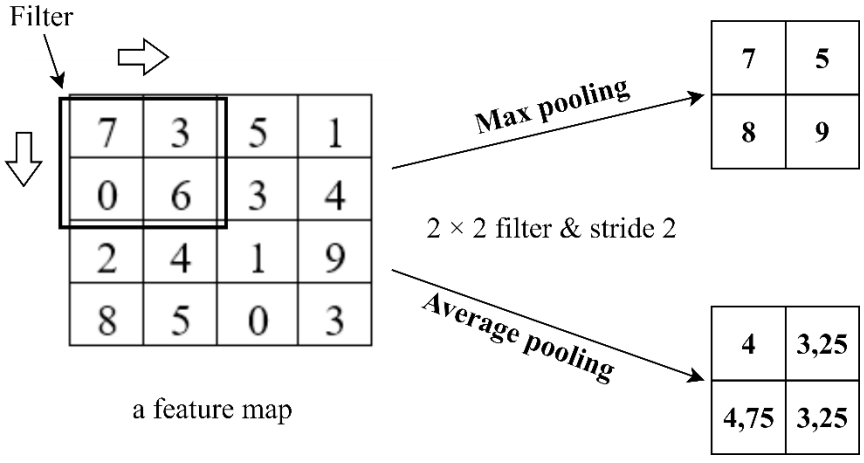


Figure 3.8 : An example of pooling layer operations.

3.3.3 Fully connected layer

The features extracted from feature maps of a model (from pooling or convolution layers) are generally flattened; in other words, the features are organized as a vector (numbers in a one-dimensional array), and then they are connected to an output layer through FCLs. A nonlinear activation function follows each of FCLs. The last FCL generally has as many nodes as the number of classes in a classification problem.

Images are fed into the model. The model extracts features automatically and, lastly, classifies images. The layers of a usual CNN are presented in Figure 3.9.

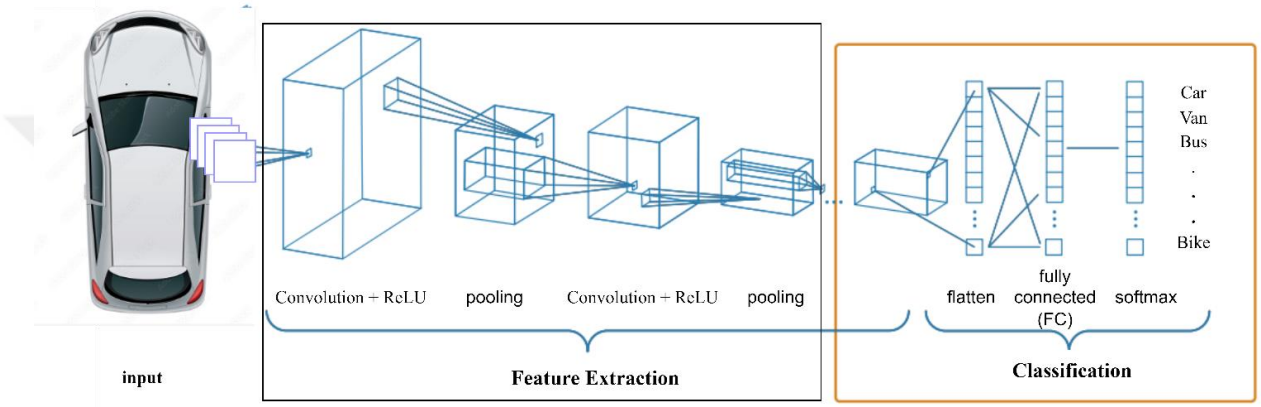


Figure 3.9 : Layers in a usual CNN.

3.3.3.1 Activation function in the last layer

An activation function is used after the last FCL, which is generally different from those used in convolution layers. The activation function type is decided according to the problem. Such as sigmoid and softmax activation functions are applied in classification problems, and identity (linear) activation function is applied for regression problems. Figure 3.10 presents the types of last layer activation functions in terms of problem type.

The softmax normalizes the outputs of the last FCL. The output of a softmax function is a vector whose values sum to one and can be commented as probabilities of target class or class membership (Yamashita et al., 2018).

The sigmoid activation function is also called a logistic function. The linear activation function is identity, its input multiplied by 1, and also called no activation.

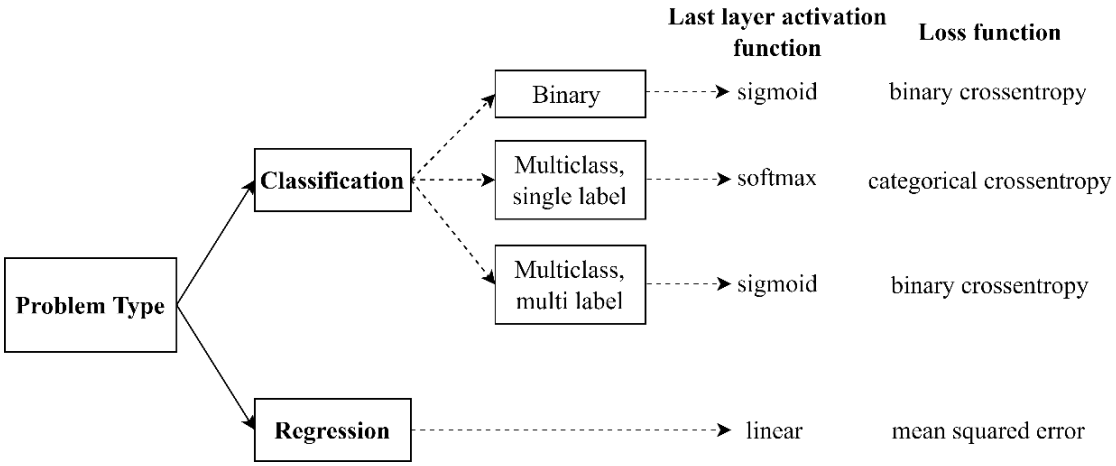


Figure 3.10 : Last layer activation function and loss function.

3.3.4 Loss function

CNNs are trained with an optimization process. The model makes predictions with a given set of weights, and the error is calculated accordingly. The loss function, error function, or cost function evaluates the compatibility between ground truth and model predictions. The choice of loss function type depends on the activation function applied in the model's output layer. Cross entropy (logarithmic) loss is a commonly used loss function in classification problems, while the mean squared error function is used in regression problems. Figure 3.10 presents loss function types (Yamashita et al., 2018).

3.3.5 On-demand layers

On-demand or secondary layers are optional; these layers can prevent overfitting and increase the generalizability of models. Batch normalization, dropout, and regularization layers are on-demand layers.

Dropout regularization eliminates some neurons with a dropout rate probability during the model training (Kandel and Castelli, 2020). For each training batch, random neurons are activated according to a predefined dropout rate. Dropout helps to build partly different networks in which the model will adjust to these variations of the network (Anaya-Isaza et al., 2021).

Batch normalization decreases the covariance shift of the network. Also, it increases the robustness of layers by adding noise. It performs by normalizing inputs of applied layers by subtracting batch mean and dividing the standard deviation of the batch. μ_A (3.3) presents mini-batch A mean, σ_A (3.4) is mini-batch A standard deviation. μ and

σ are learned. $\hat{a}^{(i)}$ (3.5) is normalized input for sample i. NA is the number of samples in the mini-batch. γ is scaling parameter, and β is shifting parameter (Kandel and Castelli, 2020). γ and β are learned, and these parameters affect the standard deviation and bias of new distribution (Anaya-Isaza et al., 2021). θ is a smoothing term. It is a tiny number that helps to prevent division by 0. $O^{(i)}$ (3.6 is the output of batch normalization.

$$\mu_A = \frac{1}{N_A}\sum_{i=1}^{N_A} x^{(i)} \tag{3.3}$$

$$\sigma_A^2 = \frac{1}{N_A}\sum_{i=1}^{N_A} (x^{(i)} - \mu_A)^2 \tag{3.4}$$

$$\hat{a}^{(i)} = \frac{a^{(i)} - \mu_A}{\sqrt{\sigma_A^2 + \theta}} \tag{3.5}$$

$$O^{(i)} = \gamma * \hat{a}^{(i)} + \beta \tag{3.6}$$

Regularization layers are often applied to limit overfitting. A penalty term is added to loss functions, preventing the usage of large weights by the model. The models with small weights offer better generalizability; it is an assumption. The weights which don't contribute to the accuracy performance of models are eliminated. The most applied ones are L1, L2, and elastic net regularizations (Kandel and Castelli, 2020).

Figure 3.11 summarizes the typical architecture and training process of a CNN. Model performance is calculated using filters and weights with a loss function by the forward propagation process on training data. The filters and weights, which are learnable parameters, are updated regarding loss value through the backpropagation process with gradient descent optimization (Yamashita et al., 2018). In a CNN, spatial hierarchies of features are learned. Initial layers learn simple or low-level features. These features are minor details of an image. Initial layers work like edge detectors and detect simple shapes, dots, lines, edges, etc., from the data. Deeper layers encode high-level, complex abstract features or higher order features, such as objects; a human nose in an image. Abstract features are built on simple features, and layers closer to the output layer interpret extracted features as part of the classification problem.

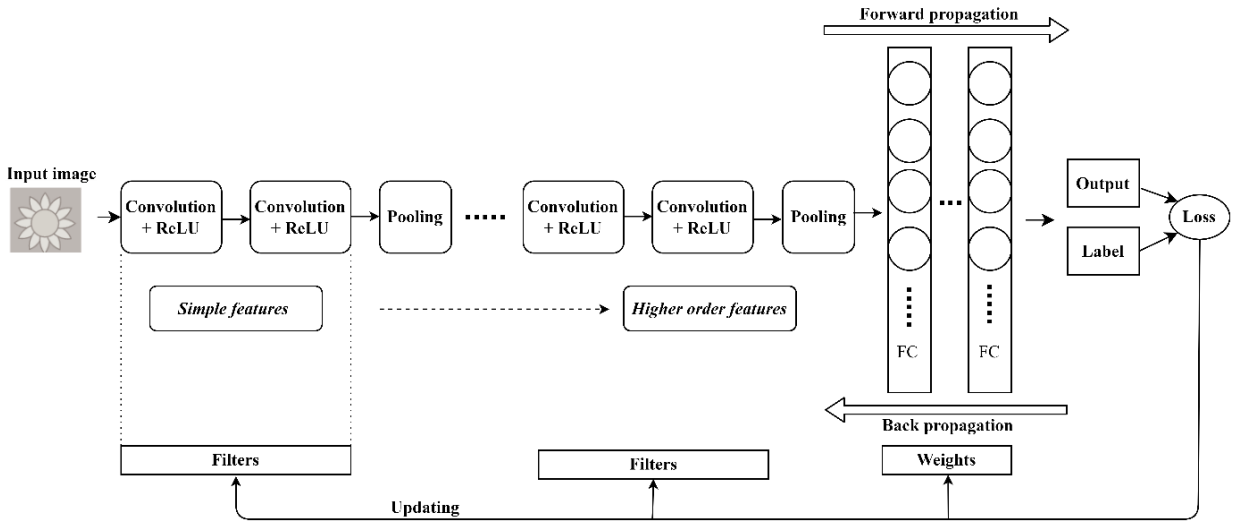


Figure 3.11 : An overview of a CNN training process.

3.4 Transfer Learning

Transfer learning is a machine learning technique, an optimization that enables speedy progress or enhanced performance for modeling another task (URL-4). In a transfer learning process, a base network is first trained on an original or base dataset for a given task. Then, learned features are repurposed or transferred to another network that will be trained on a target dataset. This process will be helpful when features are general (Yosinski et al., 2014).

Training a CNN from scratch is difficult in getting a high accuracy when you have little data to train your model. Pre-trained models can be used to increase the accuracy performance of a CNN on a different task. Pre-trained models are a common and effective approach for deep learning tasks on small image datasets. A pre-trained model is a previously trained network on a large dataset and is saved. There are some large-scale image classification tasks, such as ImageNet tasks. ImageNet is an image database. It is free data for researchers. Currently, it has more than 14,1 million images, and 21841 synsets indexed. Classes are primarily everyday objects and animals. Many pre-trained models are developed on such large and general datasets. A well-performed pre-trained network learns a spatial hierarchy of features and can become a generic model for other tasks. Features learned by pre-trained networks can be used in different computer vision tasks effectively; even a new problem may have completely different classes than the original classes. Such transfer of learned features across various tasks

is a foremost advantage of deep learning against shallow learning models. On the other hand, it makes deep learning successful for small-data tasks (Chollet, 2017).

Pre-trained networks have two uses: feature extraction and fine-tuning.

3.4.1 Feature extraction

A typical CNN consists of two parts: it starts with a series of convolution and pooling layers, called a convolutional base, and ends with a densely connected classifier, as explained earlier.

Feature extraction is made by getting the convolutional base of a pre-trained network, running the data from a new task through the convolutional base, and training a randomly initialized-new classifier on top of the output (Figure 3.9). Convolutional base is often reusable because it learns representations, and these representations are possible to be more generic. Feature maps of a CNN learn the presence of generic concepts in an image. This property makes feature maps useful in various computer vision tasks. On the other hand, representations learned by the classifier part of a CNN, will be specific to the training data on which model is trained. Densely connected layers of the classifier part give presence probability related to classes. These layers don’t contain information about objects' locations in an image. Convolutional feature maps preserve information about object locations. When object locations matter, features obtained from densely connected layers are generally useless. So reuse of the classifier part of a pre-trained model is generally avoided (Chollet, 2017).

The location or depth of convolution layers in a model affects the generality level of extracted representations. Such as layers come earlier, near the input layer in the model, extract mainly generic feature maps (textures, edges, etc.). On the other hand, layers in deeper, near fully connected layers in the model, extract more abstract feature maps, such as eye, ear, etc., on a face. So which layer to use for feature extraction depends on the similarity of datasets (Chollet, 2017).

If the original dataset on which the pre-trained model is trained is similar to new dataset, using the latter layers of the model is likely to be beneficial. Such as you have a cat versus dog classification task. You can use information learned by densely connected layers of models trained on ImageNet. Because ImageNet contains multiple cat and dog classes. Additionally, features can be extracted from the convolutional base of a pre-trained model on ImageNet. Then a new classifier can be trained on top

of these extracted features. On the other hand, when datasets are very different, i.e., classes don't overlap, layers that come earlier in the pre-trained model will be helpful instead of using the entire convolutional base for feature extraction on the new dataset (Chollet, 2017).

During the feature extraction from the convolutional base, some layers or all convolutional base are frozen, so the weights of these layers are not updated in training, which preserves representations learned. When weights of some layers in convolutional base or densely connected layers are randomly initialized, these layers' weights will be modified during training (Chollet, 2017).

Figure 3.12 presents some processes of transfer learning.

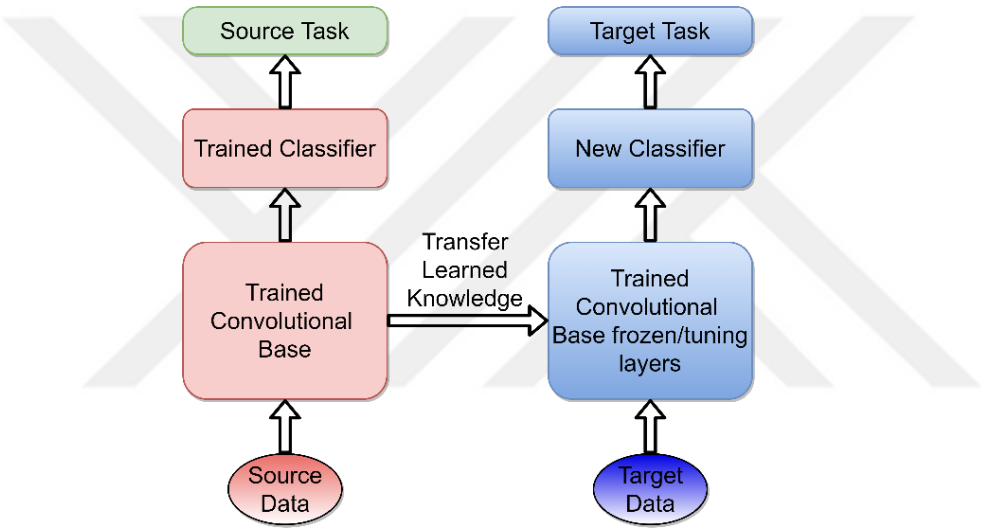


Figure 3.12 : Transfer learning process.

3.4.2 Fine-tuning

Gg Fine-tuning is a technique for model reuse and is supplementary to feature extraction. When fine-tuning is applied, some of the top layers of a frozen model are unfrozen, then these unfrozen parts: convolutional base and densely connected part, are jointly trained. Fine-tuning technique makes the model more relevant to the new problem.

The steps for an efficient fine-tuning process are as follows: a classifier part is added on top of a pre-trained base (convolutional base) network. The layers of the base network are frozen. After the classifier part is trained, some layers of the base network are unfrozen. Lastly, both of these layers are trained. When the classifier part isn't

trained first, the error will be too large during the training. Because the representations learned in advance by the layers being fine-tuned would be damaged (Chollet, 2017).

Transfer learning strategies are summarized in Figure 3.13.

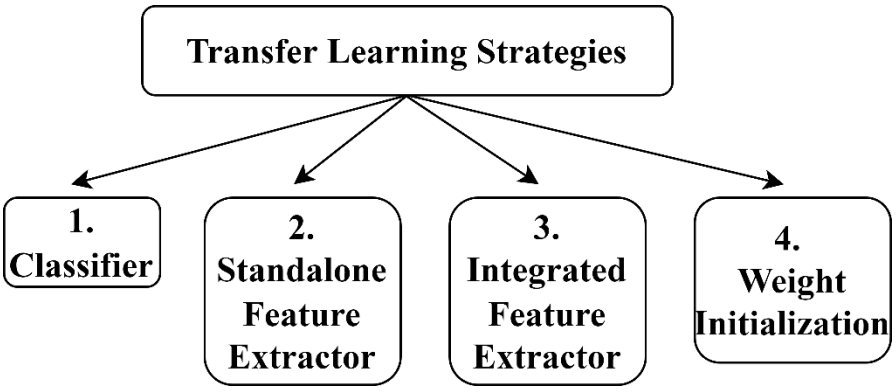


Figure 3.13 : Transfer learning strategies.

The pre-trained models can be used directly to classify new data in the first strategy, classifier. Some part of the pre-trained model, or the entire model, is used for pre-processing data and feature extraction in the second strategy; standalone feature extractor. In the third strategy, some part of the pre-trained model, or the entire model, is integrated with a new model. However, pre-trained model layers are frozen as long as the training. In the last strategy, some part of the pre-trained model or the entire model is integrated with a new model, and selected layers of the pre-trained model are unfrozen; these are trained with a new model (URL-4).

4. LITERATURE REVIEW

Obtaining handcrafted features from data is a time-consuming and challenging task. It requires expert knowledge, and this process is highly sensitive. Detecting DR by using algorithms has increased over the last few years. Recent developments in pattern recognition and artificial intelligence have made deep learning studies popular in medical fields (Bodapati et al., 2021).

Deep learning-based models are capable of automatic feature extraction and learning abstract representations. These models have achieved good results in many tasks. Compared to the techniques used in advance, deep learning models decrease the needed workforce considerably for feature extraction. So these models have overreached classic manual DR detection techniques. On the other hand, the recent developments in digital imaging and computing power have resulted in revolutions in algorithms. A massive amount of medical data is produced parallel to hardware capabilities (Farooq et al., 2022).

This section summarizes an overview of deep learning-based searches for diagnosis or severity classification of DR.

Gulshan et al. (2016) applied the InceptionV3 model by benefiting transfer learning technique for detecting RDR. The authors used EyePACS dataset in the development stage and Messidor-2 and EyePACS-1 datasets for validation. Sensitivity and specificity results are used as performance measures. The algorithm gives 97.5% sensitivity and 93.4% specificity for EyePACS-1; 96.1% sensitivity and 93.9% specificity for Messidor-2 in operating point with high sensitivity.

Vo and Verma (2016) proposed two deep CNNs for the DR recognition task. The first model is VGGNet with an extra kernel (VNXK). It has some features of GoogLeNet and ResNet in addition to VGGNet. The second model is Combined Kernels with Multiple Losses (CKML) Net. It is a modified version of GoogLeNet. Transfer learning is used to help data imbalance problems in the dataset. A hybrid color space is applied for training proposed Nets. The authors did experiments on Messidor and

EyePACS. In the normal/abnormal classification of Messidor, an accuracy of 87.1% is obtained from VNXK. In the referable/nonreferable classification on Messidor, an accuracy of 89.7% is obtained from CKML Net. In the 5-ary classification on EyePACS, VNXK is better than CKML Net, and it gives an accuracy of 83.63%.

Li et al. (2017) used four pre-trained deep CNNs, namely AlexNet, VGG-s, VGG-vd-16 & 19, in the binary classification of DR. The authors compared results from three transfer learning techniques: full fine-tuning of layers, fine-tuning the deep CNN in the layer-wise method, and extracting features from the model without tuning the network and classifying them with SVMs. According to experiment results, fine-tuning the pre-trained CNN model gives satisfactory performance when a limited number of data is available. Max accuracy of 92.01 is obtained from full fine-tuning and layer-wise tuning for the Messidor dataset; max accuracy of 94.52% is obtained from layer-wise fine-tuning for the DR dataset.

Mohammadian et al. (2017) used pre-trained CNN models: Xception and InceptionV3 in DR screening. The authors conducted experiments on the Kaggle dataset, which has 35126 images of the retina. The CNN models are fine-tuned. When the last two blocks of models are trained (fully connected layers included), 87,12% accuracy is obtained from InceptionV3, and 78,60% accuracy is obtained from Xception.

Takahashi et al. (2017) used a private dataset consisting of 9939 images, and 95% of the dataset is used in training. The authors trained a modified GoogLeNet network by changing the top layers of the network and some other parameters. The images are classified into four categories: no DR, simple DR, pre-proliferative DR, and proliferative DR. Mean accuracy of 80% is obtained from k-fold cross-validation.

Gargeya and Leng (2017) developed a data-driven deep learning model to diagnose DR. Data augmentation and pre-processing methods such as brightness adjustment, contrast adjustment, and downsizing of images are applied. The authors proposed a customized deep CNN for feature extraction from images. The extracted features are appended with metadata information to increase the diagnostic performance of prediction. A gradient boosting classifier is applied to make the final diagnosis. 75137 images are used to train and test the proposed model. Additionally, E-Ophtha and Messidor-2 datasets are used for validation. Binary classification is made. Experiment results reveal that an AUC of 97% with 98% specificity and 94% sensitivity is

achieved on the EyePACS dataset, where 5-fold cross-validation is made. An AUC of 94% and 95% is achieved on Messidor-2 and E-Ophtha datasets, respectively.

Hazim et al. (2018) used AlexNet by modifying the top fully connected layers to detect DR early. The dataset includes 580 images from Messidor. Half of the images are used in training, and the other half in testing. Binary classification is made: normal images versus images with exudates (a symptom of DR). The model gives an accuracy of 88.3%.

Lam et al. (2018) trained AlexNet and GoogLeNet networks by taking advantage of the transfer learning approach. Image normalization, contrast adjustment, and data augmentation techniques are applied. A total of 1346 images from Kaggle and Messidor datasets are used in train and validation, and a private dataset including 400 images is used in test stages. The GoogLeNet gives maximum performance and accuracy of 57.2%, 68.8%, and 74.5%, achieved from 4-ary, 3-ary, and 2-ary classifications, respectively.

Wan et al. (2018) employed VGGNet, AlexNet, ResNet, and GoogLeNet for DR detection through hyperparameter tuning and transfer learning. Kaggle dataset is used in training. 5-ary classification is made. Various evaluation criteria are used, and authors compare classification results of randomly initialized models versus hyperparameter tuning of models. According to the results, transfer learning is more advantageous than the random initialization of models. The maximum accuracy is obtained from a modified VGG, 95.68%.

Gao et al. (2018) trained deep CNN models to diagnose severities of DR. Data is collected from 3 clinical departments and is comprised of 4476 images. Pre-processing and data augmentation are applied. Besides, transfer learning is applied using pre-trained weights to initialize the networks. The authors used ResNet-18, ResNet101, VGG-19, InceptionV3 and proposed an ensemble model with four different InceptionV3. This ensemble model is called Inception@4, where different pieces of fundus images feed each InceptionV3. Then global average pooling operation is made and obtained features are concatenated. Lastly, classification is made. Inception@4 improved performance compared to InceptionV3. 5 datasets are prepared randomly for training and testing at the ratio of 4:1. The evaluation results are calculated by averaging the results from five datasets. The ensemble model, Inception@4, achieves

the best accuracy performance of 88.7% in the 4-ary classification. Severity levels are normal, moderate, heavy, and severe. Authors run models on a platform based on cloud computing.

Orlando et al. (2018) proposed a model for red lesion detection from fundus images that combines domain and deep learned knowledge. Extracted features from a patch-based CNN are increased by including manually designed features. Some morphological operations are applied to acquire possible lesion areas. Manually extracted features are obtained from each possible lesion area. These features are based on shape or intensity. The proposed CNN architecture is fed patches around possible lesion areas, and it has four convolutional layers, pooling, dropout, and one fully connected layer. Data augmentation is used in the training stage. This approach achieves satisfactory results when models are trained with lesion annotated images. A random forest classifier is used in classify hybrid features. The authors used E-Ophtha, DIARETDB1, and Messidor datasets in the experiments. The proposed model gives an AUC of 89.32% for DR screening (no DR versus any grade of DR) and an AUC of 93.47% for the need for referral (no DR and mild DR versus more severe cases of DR) on the Messidor.

Voets et al. (2019) reproduced a published study of Gulshan et al. (2016) using public data. Gulshan et al. (2016), the original research, used non-public data from EyePACS and private data for training models and Messidor -2 as a benchmark. On the other hand, Voets et al. (2019) used EyePACS data in Kaggle and another distribution of Messidor-2. The authors present difficulties in reproducing the results of a deep learning model. The model of the original study is re-implemented. An ensemble of pre-trained InceptionV3 networks is fine-tuned. The experiment results are an AUC of 95.1% on Kaggle EyePACS and 85.3% on Messidor-2. These results are far from the results of the original paper.

Sahlsten et al. (2019) used an original dataset including 41122 fundus images for the DR grading study. Additionally, the Messidor dataset is used in the validation of binary classification. The proposed model is developed based on InceptionV3. Authors conduct experiments with five different image sizes and benefit from the ensemble of deep learning models besides training a single model. The best results in the binary classification task (NRDR vs. RDR) were obtained from the largest images on Messidor, where the input size is 2095*2095 pixels. The proposed model gives an

AUC of 96.7%, sensitivity of 85.9%, specificity of 97.1%, and accuracy of 93.3% on Messidor. On the other hand, the authors obtained an accuracy of 94.4% in the 5-ary classification on the original dataset. It is the result of a training ensemble of six classifiers. The experiment results reveal that the ensemble model increased performance compared with a single model in all classification tasks, using the same image size.

Hagos and Kant (2019) used the InceptionV3 model. Features are extracted from images with pre-trained weights. The authors modified the classifier part of the model. A total of 7500 images from the Kaggle dataset are used: 2500 images for training and 5000 images for testing. Binary classification is made, and an accuracy of 90.9% and loss of 3.94% are obtained.

Zeng et al. (2019) proposed a Siamese-like CNN model to diagnose DR. Binocular images are fed into the model, enabling the use of correlation information in prediction. Model is trained using the transfer learning approach. The features are extracted from InceptionV3. The training dataset is comprised of 28104 images, and the test dataset is comprised of 7024 images. Binary classification gives an AUC of 95.1% and a sensitivity of 82.2% in diagnosing RDR. 5-ary classification gives a kappa score of 82.9% in a validation set (10%).

Zhang et al. (2019) proposed a model, namely DeepDR, for identification and grading DR. Authors benefit from ensemble learning and transfer learning. Various combinations of feature extraction models are tried to obtain an optimal result. The authors used a private dataset with 3833 fundus images: 2756 for training, 306 for validation, and 771 for testing. The training dataset is expanded using the data augmentation technique. The proposed model steps are pre-processing images, feature extraction with transfer learning, global average pooling operation, feature classification with deep NN, feature fusion, and reporting results. In the identification model (binary), Xception, InceptionV3 and InceptionResNetV2 are used in feature extraction. In the grading model (there are four grades), ResNet50, DenseNet169, and DenseNet201 are used in feature extraction. The authors proposed two customized deep NN architectures as classifiers. These deep NNs are fed from the global average pooling operation output that is applied to extracted features. The identification model gives an AUC of 97.7%, specificity of 97.7%, and sensitivity of 97.5%. The grading model gives a specificity of 98.9% and a sensitivity of 98.1%.

Kassani et al. (2019) used a modified Xception architecture in feature extraction from fundus images. The features are extracted from different convolutional layers of Xception and are aggregated. A MLP is fed with extracted features and diagnoses the severity of DR. Additionally, the authors used original Xception, MobileNet, ResNet50, and InceptionV3 as feature extractors to assess the performance of the proposed model. Transfer learning and hyper-parameter tuning are employed to increase overall classification performance. Aptos dataset is used in model evaluation. The dataset has 3662 images, and 10% of the dataset is used in the test. According to experiment results, modified Xception as a feature extractor performs better than the original Xception and the other three deep CNN models. It gives an accuracy of 83.09%, sensitivity of 88.24%, and specificity of 87% in the 5-ary classification.

Mateen et al. (2019) proposed a DR severity classification model,VGGNet, where a Gaussian mixture model for region segmentation, analysis of connected component for DR features, principle component analysis and singular value decomposition for feature selection, VGG-19 architecture for feature extraction, and lastly softmax for classification are used. Kaggle dataset is used in experiments. The maximum classification accuracy of 98.34% is obtained using singular value decomposition with fully connected layer-7 of VGGNet.

Li et al. (2019) used the InceptionV3 network by taking advantage of the transfer learning technique. A private dataset consisting of 8816 images is used in training and validation. Additionally, the Messidor-2 dataset is used in validation. Some pre-processing methods such as histogram equalization and denoising techniques are applied. Firstly, the authors randomly initialized the fully connected part on InceptionV3, and the rest of the network is frozen. Later, fine-tuning is tried by unfreezing some layers of the network, and the model is retrained until no further improvement is obtained. An accuracy of 93.49% is obtained for no referral vs. referral DR classification on Messidor-2.

Tseng et al. (2020) proposed a hybrid architecture that consists of a multi-modal DL model with lesion information. Specifically, two fusion architectures based on DL exist. These are called late fusion and two stages early fusion. The first one fuses lesion detection and classification models in parallel using some post-processing steps. The grading model and four lesion classification models are trained separately. InceptionV4 is used in grading, and DenseNet is used for lesion classification. The

extracted features from these models (grading and lesion classification) are fed into a softmax regression, and final outputs are concatenated. Lastly, disease severity is classified with ridge regression. The second architecture fuses lesion detection and classification models in order. Two types of images are used in grading. The RetinaNet (used in object detection) is trained with raw images in the first stage. Then, lesion-enhanced and raw images are fed into different InceptionV4 models, which are trained simultaneously. Lastly, both features are concatenated, and classification is made. The authors used 26699 images from 3 Taiwanese hospitals. The Messidor-2 dataset is used as a benchmark. The proposed fusion model achieved satisfactory results. The maximum accuracy of 91.09% and 85.12% are achieved in binary and 5-ary classification tasks, respectively, in the original data test set. An accuracy of 91.99% is obtained on Messidor-2 in binary classification (NRDR vs. RDR).

Bodapati et al. (2020) extracted features from fundus images using multiple pre-trained models. These models are VGG-16, NASNet, Xception, and InceptionResNetV2. Each model enables the extraction of different features from images so that the proposed model gives better representations. The models are blended through a fusion process. The fusion is based on pooling and cross-pooling operations on extracted features. A DNN model is trained to classify obtained features. The authors used the Aptos dataset in experiments. The most proper feature fusion for DR diagnosis is achieved with the fusion of features from VGG-16 and Xception. The experiment results of the proposed model (classifying blended features with DNN) are an accuracy of 97.92% for DR identification and an accuracy of 80.96% for severity prediction.

Doshi et al. (2020) applied various downscaling algorithms to cope with different sizes of images before using deep learning models. These algorithms are the nearest neighbor, bicubic, bilinear, Lanczos, rapid detail image preserving, and learned downscaling image algorithms. The authors combined Indian DR and Kaggle datasets. A multi-channel architecture is proposed where InceptionV3 is used in feature extraction. The pre-trained weights are used. The pre-processed images are divided into four pieces, and each piece is fed into a different InceptionV3 (feeding inputs to a multi-channel network). The InceptionV3s are trained in parallel. Extracted features from four InceptionV3 are concatenated. Then fully connected layers are trained to make a binary classification. The maximum accuracy performance is achieved from

multi-channel InceptionV3 with a learned image downscaling algorithm. The network gives an accuracy of %85.2, specificity of 87.6%, and 83.4% sensitivity.

de La Torre et al. (2020) proposed a DL interpretable classifier model for DR. The classifier explains the results by presenting pixel contributions. A score for every point in input and hidden spaces is assigned. The pixel-wise scores are propagated in the model. The visual maps are generated to help diagnose DR. The EyePACS dataset on Kaggle is used in experiments, and Messidor-2 is used as a benchmark. Data augmentation is applied because of data imbalance. The customized CNN model has 391125 parameters and consists of seventeen layers. Some layers take care of feature extraction, and the rest are used for classification. The authors made a binary classification: one class for the severe level of DR (classes 2, 3 and macular edema) and the other for non-sever cases (no DR and class 1). A sensitivity of 90.6%, specificity of 84.7%, an accuracy of 85.7%, and an F1 score of 71% are obtained on the test set of EyePACS. On the other hand, a sensitivity of 90.8%, specificity of 91.1%, an accuracy of 91.0%, and an F1 score of 89.6% are obtained on Messidor-2.

Toledo-Cortés et al. (2020) proposed a hybrid model composed of DL and Gaussian process for DR diagnosis. The authors also search the effect of uncertainty quantification on model performance. The proposed model has three stages. The first stage is pre-processing of images. Then features are extracted from an InceptionV3. EyePACS dataset is used in fine-tuning the model. The last stage is classifying obtained features with a Gaussian Process Regressor, a Bayesian regression approach. The Messidor-2 dataset is used in the test. Data augmentation is applied. Binary classification, namely NRDR vs. RDR, is made. A sensitivity of 72.37%, a specificity of 86.25%, and an AUC of 87.87 are achieved on Messidor-2.

Saxena et al. (2020) applied CNN-based deep learning models for DR detection. The pre-processed and augmented images are fed into InceptionV3 or InceptionResNetV2 models. Binary classification is made: no-DR vs. any grade of DR. The EyePACS dataset is used to train and test the models. Additionally, benchmark datasets of Messidor and Messidor-2 are used in the test. The authors benefit from the ensemble technique. Multiple models are trained, and their outcomes are combined through an ensemble. Extracted features are classified with Extreme Gradient Boosting, Artificial Neural Networks, and SVM. Neural networks performed better than the others. The best results are obtained when InceptionResNetV2 is used as a feature extractor. The

proposed model acquires an AUC of 92% with specificity and sensitivity of 86.09% and 81.02%, respectively, on Messidor-2; an AUC of 95.8% with specificity and sensitivity of 89.92% and 88.84% on Messidor.

Zago et al. (2020) proposed a lesion localization model that takes advantage of a patch-based CNN for DR diagnosis. The authors aim to classify each patch from a given fundus image into non-lesion or lesion clusters. A five-layer CNN is applied to classify patches of training data; 50000 non-lesion and all lesion patches are selected using some criteria. The selection process enables the challenging images to be given particular attention while training. The VGG-16 model is applied as follows: pre-trained weights are used in the initialization of the model, and then the model is tuned with the selected patches. The overall training is patch-based. The classification is based on the probability of the lesion seen in the patch. The authors trained the model on the DIARETDB1 database, which has labels for some lesion types. The model is validated on the DIARETDB0, Messidor, Messidor-2, Kaggle, DDR, and IDRiD datasets. The proposed model achieves an AUC of 91.2% and a sensitivity of 94% for no-DR vs. any grade of DR classification; an AUC of 93.6%, and a sensitivity of 97.6% for no-DR or mild DR vs. all others using the Messidor dataset; an AUC of 94.4%, a sensitivity of 90%, and a specificity of 87% in detecting RDR using the Messidor-2 dataset.

Tymchenko et al. (2020) built a DR classification model using encoder and decoder structures. The features are extracted from the encoder with pre-trained CNN models. The extracted features are fed into three decoders that are called heads. These are classification, regression, and ordinal regression heads. Lastly, a linear regression makes the final prediction from the outputs of heads. The authors used an ensemble of SE-ResNeXt50, EfficientNet-B4, and EfficientNet-B5 architectures.

Das et al. (2021) developed a deep learning model that uses pre-processed and segmented retinal images. Branching of blood vessels is extracted, and segmented areas are evaluated by applying morphological operations and histogram equalization technique. The authors used a customized CNN architecture that includes the memory and central CNN modules. The memory module has excitation, squeeze, and bottleneck layers, which decrease model complexity and enables feature extraction. The other module has an ordinary convolutional layer structure. Both modules make

the classification. DIARETDB1 dataset is used in experiments, and the model gives an accuracy of 98.7% for binary classification.

Ayala et al. (2021) proposed a DenseNet-121-based model. The transfer learning approach is used in the optimization of model parameters. Messidor-2 and Aptos datasets are used, and accuracy of 64% and 81% are obtained in 5-grade classification, respectively.

Tariq et al. (2021) applied GoogleNet, AlexNet, InceptionV4, ResNeXt50, and InceptionResNetV2 networks by taking advantage of transfer learning. A custom dataset is prepared from different resources. Data augmentation is applied. The maximum accuracy of 97.53% is obtained from the ResNeXt50 model for the 5-grade classification task.

Paradisa et al. (2021) proposed a concatenate model. The features are extracted from InceptionResNetV2 and DenseNet121 networks and combined. A MLP classified the extracted features. DDR dataset is used, and three grade classification is made. The model achieves 91% accuracy.

Yi et al. (2021) proposed a network composed of EfficientNet and residual attention block. The residual attention block is added to the output of EfficientNet to extract more valuable features and better distinguish between lesions. A transfer learning approach is used. The output obtained from the attention block is sent to classifiers. Aptos dataset is used, which has 3662 images. The EyePACS dataset is also used for benchmarking. The proposed model achieves an accuracy of 98.36% for binary classification; an accuracy of 93.55% for 5-grade classification on the Aptos dataset.

Bodapati et al. (2021) proposed a hybrid approach combining deep CNNs and a gated attention mechanism for DR classification. The features are extracted from pre-trained models. The gated attention mechanisms give more attention to lesion parts of retinal images than non-lesion parts. The authors used the Aptos dataset that is available on Kaggle. The proposed model provides an accuracy of 82.54% for DR severity (5-grade) classification. VGG-16 and Xception networks are used in feature extraction.

Imran et al. (2022) applied a new entropy enhancement method to increase the visibility of images. The authors proposed a hybrid network that consists of main and auxiliary models. The main model includes residual and transition blocks as well ordinary CNN layers. The auxiliary model has layers of convolution, pooling, and

flattening. This model takes advantage of complementing the main model's performance. The extracted features from both models are concatenated and fed into the classifier. Three datasets are used: UWF, Aptos, and Messidor-2. 5- grade classification is made, and the proposed network gives an accuracy of 84% on UWF, 80% on Messidor-2, and 92% on the Aptos dataset.

Nneji et al. (2022) proposed a weighted fusion model using deep learning. Images are pre-processed using histogram equalization and edge detection methods. InceptionV3 and VGG-16 are fine-tuned, and some modifications are made to extract features from processed images. The extracted features are combined with weighting them, and a softmax classifier is applied. Data augmentation is carried out. The proposed network gives an accuracy of 98.5% for four-grade classification on Messidor and an accuracy of 98% for five-grade classification on the EyePACS dataset.

Ali et al. (2022) proposed a customized CNN architecture that employs parallel and series of small network modules. These modules have convolutional and classification layer blocks and are computationally efficient. The maximum accuracy of 97.9% and AUC of 99.6% are achieved on the Aptos dataset in binary classification.

Table 4.1 lists reviewed studies using CNN architectures to classify DR and summarizes the applied CNN model, dataset, number of classes, and performance measure of studies. Transfer learning is performed in most of the listed studies.

Table 4.1 : The studies applying CNN models.

Study	Model	Dataset (number of images)	# of classes	Performance Measure
Gulshan et al. (2016)	InceptionV3	EyePACS (128175), Messidor-2 (1748), EyePACS-1 (9963)	2 (RDR vs. NRDR)	SE, SP
Vo and Verma (2016)	Modified VGG, Modified GoogLeNet	EyePACS (35126), Messidor(1200)	5 for EyePACS, 2 (DR vs. DR and Referable vs. Nonreferable) for Messidor	Acc SE, SP, ROC
Li et al. (2017)	AlexNet, three VGG-based architectures	DR1 (1014), Messidor (1200)	2 (no DR vs. any grade of DR)	Acc, SE, SP, AUC
Mohammadian et al. (2017)	Xception InceptionV3	Kaggle (35126)	2 (no DR vs. any grade of DR)	Acc
Takahashi et al. (2017)	Modified GoogLeNet	Private (9939)	4 (NDR, SDR, PPDR, PDR)	Acc

Table 4.1 (continued) : The studies applying CNN models.

Study	Model	Dataset (number of images)	# of classes	Performance Measure
Gargeya and Leng (2017)	customized CNN	EyePACS (75137), Messidor-2 (), E-Ophtha ()	2 (no DR vs. any grade of DR)	SE, SP, AUC
Hazim et al. (2018)	AlexNet	Messidor (580)	2 (normal vs exudates images)	Acc
Lam et al. (2018)	AlexNet GoogLeNet	Kaggle + Messidor (1346), Private (400)	2 (normal or mild vs. moderate to severe stages), 3 (no DR, mild, severe), 4 (grades according to lesion descriptions)	Acc
Wan et al. (2018)	various VGGNets, AlexNet, ResNet, GoogLeNet	Kaggle (35126)	5 (no DR, mild, moderate, severe, proliferative)	Acc, SE, SP, AUC
Gao et al. (2018)	ResNet-18, ResNet-101, VGG-19, InceptionV3, Inception@4	Private (4476)	4 (no DR, moderate, heavy, severe)	Acc, PR, Rec
Orlando et al. (2018)	customized CNN	E-Ophtha (381), DIARETDB1 (89), Messidor (1200)	2 (no DR vs. any grade of DR), 4 (no DR and mild Dr vs. more severe case of DR)	SE, AUC
Sahlsten et al. (2019)	InceptionV3, ensemble of models	Private (41122), Messidor (1200)	2 (RDR vs. NRDR), 5 (no DR, mild, moderate, severe, proliferative)	Acc, SE, SP, AUC, Macro-AUC
Hagos and Kant (2019)	InceptionV3	Kaggle (7500)	2 (no DR vs. any grade of DR)	Acc, Loss
Zeng et al. (2019)	InceptionV3	Kaggle (35128)	2 (RDR vs. NRDR)	SE, SP, AUC, Kappa
Mateen et al. (2019)	VGG-19	Kaggle (35126)	5 (no DR, mild, moderate, severe, proliferative)	Acc
Zhang et al. (2019)	Xception, InceptionV3, InceptionResNetV2, ResNet50, DenseNet169, DenseNet201	Private (3833)	2 (no DR vs. any grade of DR), 4 (no DR, NPDR, NPDR2PDR, PDR)	Acc, SE, SP, PR AUC, Kappa, F1-score, Fβ-score, Youden's index
Kassani et al. (2019)	Modified Xception, Xception, InceptionV3, ResNet50, MobileNet	Aptos (3662)	5 (no DR, mild, moderate, severe, proliferative)	Acc, SE, SP

Table 4.1 (continued) : The studies applying CNN models.

Study	Model	Dataset (number of images)	# of classes	Performance Measure
Li et al. (2019)	InceptionV3	Private (8816), Messidor (800)	2 (RDR vs. NRDR), 5 (no DR, mild, moderate, severe, proliferative)	Acc, SE, SP, AUC
Tseng et al. (2020)	DenseNet121, InceptionV4	Private (26699), Messidor-2 (1748)	5 (no DR, mild, moderate, severe, proliferative), 2 (RDR vs. NRDR)	Acc, SE, SP, AUC, weighted k statistic
Bodapati et al. (2020)	NASNet, VGG-16, Xception, InceptionResNetV2	Aptos (3662)	2 (no DR vs. any grade of DR), 5 (no DR, mild, moderate, severe, proliferative)	Acc, Kappa
Doshi et al. (2020)	InceptionV3	IDRID (516), Kaggle (35126)	2 (no DR vs. any grade of DR)	Acc, SE, SP
de La Torre et al. (2020)	customized CNN	EyePACS (88650), Messidor-2 (1748)	2 (the most severe cases of DR vs. the rest)	Acc, SE, SP, F1-score, Matthews correlation coefficient, positive predictive value, negative predictive value
Toledo-Cortés et al. (2020)	InceptionV3	EyePACS (65617), Messidor-2 (1748)	2 (NRDR vs. RDR)	Acc, SE, SP
Saxena et al. (2020)	InceptionV3 and InceptionResNetV2	EyePACS (56839), Messidor-2 (1748), Messidor (1200)	2 (no DR vs. any grade of DR)	SE, SP, F1-score, Rec, AUC
Zago et al. (2020)	VGG-16	DIARETDB0 (130), DIARETDB1 (89), Messidor (1200), Messidor-2 (874), Kaggle (15919), DDR (4105), IDRiD (103)	2 (no DR vs. any grade of DR), 2 (no DR or mild DR vs. all others), 2 (NRDR vs. RDR)	SE, SP, AUC

Table: 4.1 (continued) : The studies applying CNN models.

Study	Model	Dataset (number of images)	# of classes	Performance Measure
Tymchenko et al. (2020)	Ensemble of EfficientNet-B4 & -B5, SE-ResNeXt50	Kaggle (35126), IDRID (413), Messidor (1200), Aptos (13000)	2 (no DR vs. any grade of DR)	Acc, SE, SP, quadratic weighted kappa, Macro F1
Das et al. (2021)	customized CNN	DIARETDB1 (89)	2 (no DR vs. any grade of DR)	Acc, SP, PR, Rec
Tariq et al. (2021)	GoogleNet, AlexNet, InceptionV4, ResNeXt50, and InceptionResNetV2	Private (5333)	5 (no DR, mild, moderate, severe, proliferative)	Acc, SP, PR, Rec, F1-score
Ayala et al. (2021)	DenseNet121	Aptos (3662), Messidor-2(1744)	2 (no DR vs. any grade of DR); 5 (no DR, mild, moderate, severe, proliferative)	Acc, PR, Rec, F1-score
Paradisa et al. (2021)	InceptionResNetV2, DenseNet121	DDR (2739)	3 (no DR, NPDR, PDR)	Acc, PR, Rec, F1-score
Yi et al. (2021)	EfficientNet with a Residual Attention Block	Aptos (3662), EyePACS (11756)	2(no DR vs. any grade of DR); 5 (no DR, mild, moderate, severe, proliferative)	Acc, PR, F1-score, Kappa
Bodapati et al. (2021)	VGG-16, Xception, Gated Attention Blocks	Aptos (3661)	5 (no DR, mild, moderate, severe, proliferative)	Acc, PR, Rec, F1-score, Kappa
Imran et al. (2022)	customized CNN	UWF (1600), Aptos (3662), Messidor-2 (1748)	5 (no DR, mild, moderate, severe, proliferative)	Acc, SE, SP, PR, PRC, AUC
Nneji et al. (2022)	Modified InceptionV3 and VGG-16	Messidor (1200), EyePACS (2000)	4 (no DR, mild, moderate or severe, proliferative); 5 (no DR, mild, moderate, severe, proliferative)	Acc, SE, SP, PR, AUC
Ali et al. (2022)	customized CNN	Aptos (3662)	2 (no DR vs. any grade of DR)	Balanced Acc, SE, SP, AUC, testing time

Acc: Accuracy, SE: Sensitivity, SP: Specificity, Rec: Recall, PR: Precision, AUC: Area under the curve, PRC: Precision and Recall Curves

Feature selection is a process of selecting contributing features for relative tasks. It typically has two steps: searching feature subsets and evaluating subsets to choose the best among features. Metaheuristic algorithms are applied to search feature subsets (Meenachi and Ramakrishnan, 2021).

Gunavathi and Premalatha (2015) applied the Cuckoo Search optimization algorithm for feature selection from gene expression data, which is used in cancer classification. Feature selection is critical in gene expression datasets because of its high dimension.

Before feature selection, genes are ranked with some statistical analyses. The authors experimented with the proposed model with different cancer datasets.

Darwish et al. (2018) used swarm-based algorithms, namely grey wolf optimizer, whale optimization, moth flame optimization, and flower pollination algorithms for feature selection, and then selected features are classified. Results reveal that these algorithms are practical for feature selection from breast cancer data.

Meenachi and Ramakrishnan (2021) proposed two hybrid feature selection algorithms for the classification of cancer. Authors used tabu search, ant colony optimization, genetic algorithm, and fuzzy rough set in building optimal feature selection algorithms. Five datasets are used, and the result confirms that the proposed algorithms successfully select local and global features.

Uzer et al. (2013) used the Artificial Bee Colony algorithm in feature selection, and selected features are classified with SVM algorithm. The authors used different medical datasets: liver disorders, hepatitis, and diabetes.

Wang et al. (2014) used six metaheuristic algorithms, namely particle swarm optimization, genetic algorithm, ant colony optimization, quantum inspired evolutionary algorithm, differential evolution, and harmony search for biomedicine feature selection problems, which are high dimensional problems. Authors designed ten benchmark problems where dimensions range from 200 to 20,000. To verify algorithms, correlated, redundant, useless, relevant, and misleading features are included in datasets. According to the results, all metaheuristic algorithms are satisfactory in low dimensional (less than 200 dimensions) feature selection problems. When the dimension number increases, the performance of metaheuristics changes muchly. Modified differential evolution is proposed for feature selection in high-dimensional problems.

Canayaz (2021) applied particle swarm optimization and gray wolf optimization for feature selection from extracted features of several deep learning models. The author used X-ray images for the Covid-19 diagnosis.

Lin et al. (2021) proposed two variants of the ABC algorithm for feature (molecular descriptors) selection in quantitative structure-activity relationship problems where the goal is to build connection between molecular structure attributes of chemical compounds and their biological activities. The comparison results show that one of the

proposed ABC algorithms outperforms other algorithms regarding prediction performance, number of selected feature sizes, and stability.

This study applies three metaheuristic algorithms in the feature selection process.

5. METHODOLOGY

5.1 Dataset

In this study, a publicly available dataset, Messidor-2, is used. The Messidor stands for Methods to Evaluate Segmentation and Indexing Techniques in Retinal Ophthalmology".

The first Messidor dataset (original) has been provided to help researchers develop computer-assisted DR diagnosis models. The original dataset contains 1200 images of DR examination. On the other hand, Messidor-2 is an extensive dataset of the original one and contains 1748 images (874 examinations, images in pairs). The dataset is provided by program partners, including some hospitals, a university, ophthalmology laboratories, and some other providers.

Images are macula-centered eye images, also called fundus images. They are in PNG and JPG formats. There are five adjudicated grade levels: proliferative, severe, moderate, mild, and no-DR. Some third parties, except Messidor program partners, proposed annotations for images in the dataset. These annotations don't present the ground truth of DR grade levels. Seven images are excluded because of some missing information or image quality. So 1741 images are used in the analysis.

A color image has multiple channels: red, green, and blue, abbreviated as RGB. So when a colorful image is given as an input to a model, three images (R, G, B channels) are given from a data perspective.

This study categorizes the dataset regarding international clinical DR and macular edema disease severity scales (PIRC and PIMEC). A binary classification task is carried out. Class names are no referable DR (NRDR) and referable DR (RDR). NRDR class contains cases of mild DR and no DR. On the other hand, RDR class contains the cases of moderate, severe, and proliferative DR. The distribution of relabeled images is presented in Table 5.1. This binary classification is famous in recent DR classification studies (Sahlsten et al., 2019). The NRDR class has 1286 images and is labeled with "0". The RDR class has 455 images and is labeled with "1".

Table 5.1 : The distribution of relabeled data.

Label	Class name	Number of images
0	NRDR	1286
1	RDR	455

More details on the Messidor-2 dataset can be found on the website and in referenced articles (URL-5; Decencière et al., 2014; Abràmoff et al., 2013).

5.2 Proposed Model

The proposed model is given in Figure 5.1. The model has four steps. The first step is pre-processing of images. To extract abstract features, the pre-processed images are fed into a pre-trained CNN architecture, InceptionV3. The transfer learning method is applied. A pooling operation is applied to extracted features, and a feature vector is obtained in the second step.

Three metaheuristic algorithms (MHAs): Particle Swarm Optimization, Simulated Annealing, and Artificial Bee Colony are used for feature selection from previously obtained feature vector in the third step. The feature selection process enables us to obtain the best potential features.

Lastly, machine learning algorithms are applied to classify extracted and selected features as RDR versus NRDR. A boosting model (XGBoost), bagging models (bagged decision trees, random forest, extra trees), SVM, a linear classifier (logistic regression), and a neural network model (MLP) are classifiers.

The details of the steps are explained in the following sections.

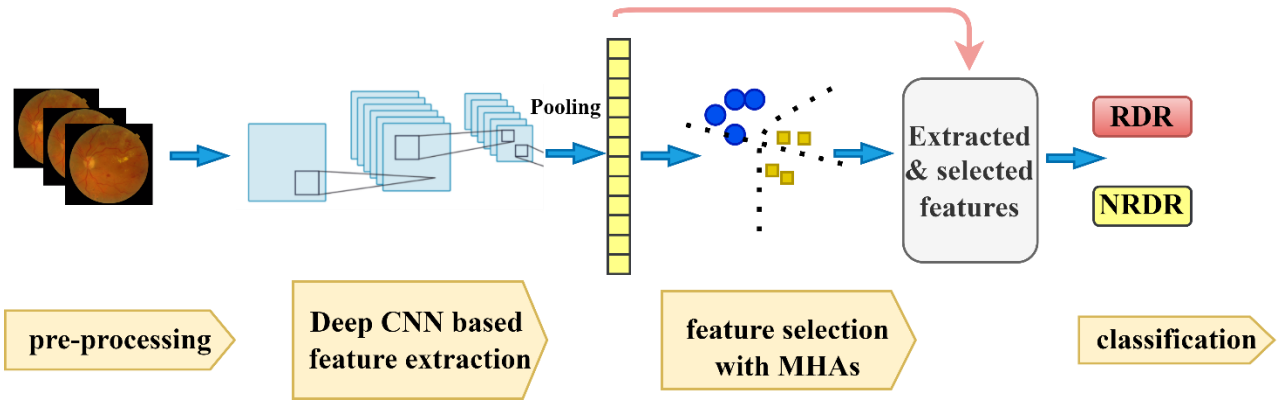


Figure 5.1 : The proposed model.

5.3 Pre-processing

The resolution of fundus images varies in the dataset. Such as there are the resolutions of 1440×960, 2304×1536, or 2240×1488. As the first pre-processing operation, images are resized to a resolution of 1200 (width)×960 (height).

Secondly, pixel normalization is applied. Image pixel values are rescaled from the range of [0-255] to the range of [0-1] in the RGB channels. The obtained images are fed into the following process, feature extraction.

5.4 Feature Extraction

Deep learning networks require vast amounts of data to learn the latent patterns in data successfully. For some problems, it is not easy to collect large-scale datasets. On the other hand, labeling the collected data could be costly in terms of taking time, the insufficient number of experts, etc. Medical tasks are one area where limited data and costly labeling challenge developing well-performing deep learning models. The transfer learning approach helps to solve problems with an insufficient number of training data. Besides, training and testing data are not necessary to be independent and identically distributed. The model of the target task doesn't require to be trained from scratch, which initializes network weights randomly (Tan et al., 2018). A pre-trained network can be used as a starting point to learn a problem in a different domain. Learned features can be transferred to new problems.

There are deep learning models that are made available with pre-trained weights. These pre-trained models can be used for various tasks such as feature extraction, fine-tuning, or prediction. Keras (URL-6) made accessible many pre-trained models that are performing well on ImageNet tasks. Such as Xception, VGG-19, ResNet152, MobileNet, DenseNet201, NASNetMobile, EfficientNetB7, InceptionV3, InceptionResNetV2.

ImageNet is a dataset of images. It is organized in a hierarchy of nodes that include thousands of images. ImageNet is a project for advancing deep learning and computer vision research. Because it is critical to have a large-scale image dataset for designing more generalizable and sophisticated models. On the other hand, ImageNet tries to satisfy the demand for a high-quality object categorization benchmark with evaluation metrics. The data is serviced for free. The most used dataset is the ImageNet large-

scale visual recognition challenge (ILSVRC) 2012-2017 image localization and classification dataset. This dataset includes 1,281,167 training, 50,000 validation, and 100,000 test images in 1000 object classes. ILSVRC has helped with benchmarking and developing many state-of-the-art algorithms (URL-7).

There are several versions of the inception network, which have iterative advancements over the initial one, such as InceptionV1 (Szegedy et al., 2015), InceptionV2 (Szegedy et al., 2016), InceptionV3 (Szegedy et al., 2016), or Inception-ResNet (Szegedy et al., 2017) networks. Researchers tried to increase the performance of traditional CNN models by building deeper networks. Deep networks tend to overfit; on the other hand, the training of such networks brings gradient updates and computational difficulties. Choosing the suitable kernel sizes for CNN layers is challenging because of the high variance in position information in images. Inceptions networks have developed to satisfy these issues in the general sense.

In this study, pre-trained weights of the InceptionV3 network (Szegedy et al., 2016) on ImageNet are used for the feature extraction from fundus images. InceptionV3 is a widely applied network in image classification tasks.

The Inception concept is developed from the GoogleNet network and seen in ILSVRC of 2014. It is inspired by primates' visual cortex, enabling them to cope with multiple scales. It has a network-in-network approach, which increases the representational power of the network and offers computational advantages. Developers concentrated on optimizing the network's performance by increasing depth and size (the traditional way) and using sparsity in layers based on Arora et al. (2014). Similar sparse nodes were clustered into a dense structure. Also, they built inception modules taking advantage of visual information multiscale analysis in 1×1, 3×3, and 5×5 convolution layers. After that, these layers were connected through dimension reduction to result in 1×1 convolutions (Nivrito et al., 2016; Li et al., 2019).

Arora et al. (2014) suggested a layer-by-layer construction based on analyzing the last layer's correlation statistics, grouping (clustering) the highly correlated units, and then fed-forwarding the units to the next layer. Developers suppose that each unit of the previous layer matches up with some region in the input image. The units are classified into filter banks. Correlated units are concentrated in local regions in the layers close to input. It means the network would result in many clusters concentrated on only a

region, and a layer of 1×1 convolutions can cover them in the next layer. On the other hand, there can be a less number of clusters that spread out more spatially. Larger patches of convolutions can cover these clusters. Meanwhile, there will be a diminishing number of patches over larger regions. The filter sizes 1×1, 3×3, and 5×5 are chosen to avoid patch alignment issues in inception architecture. Since pooling operations have successfully performed many CNN tasks, the pooling layer is added in the inception module. These layers are ordered on the same level rather than stacked sequentially. It makes the network wider (Szegedy et al., 2015).

The naïve version of the inception module is presented in Figure 5.2. Using this naïve version of the module brings problems with the computational expense. The merging of outputs from convolutions and pooling operation would reason an imperative rise in output number from phase to phase (Szegedy et al., 2015). This problem leads to another idea for the proposed module.

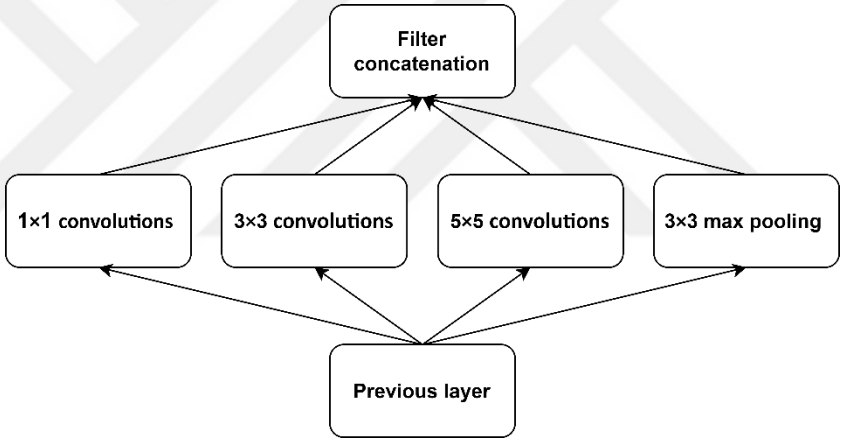


Figure 5.2 : Naïve version of inception module (Szegedy et al., 2015).

1×1 convolution is added before 5×5 and 3×3 convolution layers. Thus, the depth of tensor (the number of input channels) decreases before 5×5 and 3×3 convolution layers, which helps to decrease computational expense. The obtained outputs are concatenated and sent to the next layer (URL-8).

The inception module with dimension reductions is presented in Figure 5.3.

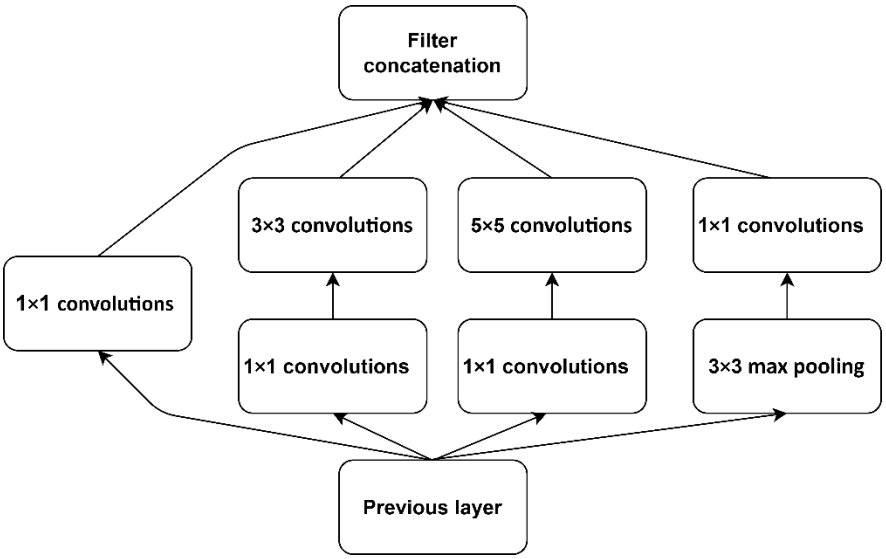


Figure 5.3 : An inception module with dimension reductions (Szegedy et al., 2015).

Extensive dimension reduction by inception modules helps preserve a large number of input filters from the previous layer to the next layer. Procession of the visual information in multiple scales and then aggregation in filter concatenation enables the next layer to abstract features from various scales synchronously (Szegedy et al., 2015).

InceptionV2 and InceptionV3 proposed modules that improve accuracy and computational performance (Szegedy et al., 2016). InceptionV2 has inception modules where two 3×3 convolutions are applied instead of a 5×5 convolution. This factorization into smaller convolutions increases computational performance. Because convolution operation with large spatial filters is expensive in computation, such as a 3×3 convolution operation with m filters is about 2.77 (25/9) times computationally cheaper than a 5×5 convolution operation with the same number of filters.

The factorization method is illustrated in Figure 5.4.

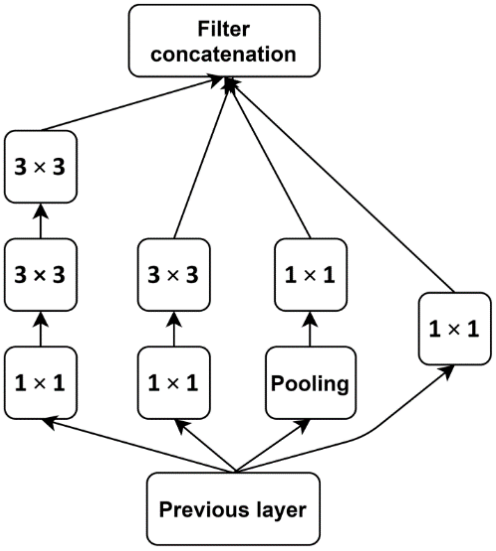


Figure 5.4 : The inception module of two 3×3 convolutions replaces 5×5 convolutions (Szegedy et al., 2016).

In addition, the authors used a combination of 1×m and m×1 instead of m×m convolution. Such as, applying a 1×5 convolution followed by a 5×1 convolution is equivalent to applying a 5×5 convolution. Moreover, this operation is cheaper in terms of computational cost.

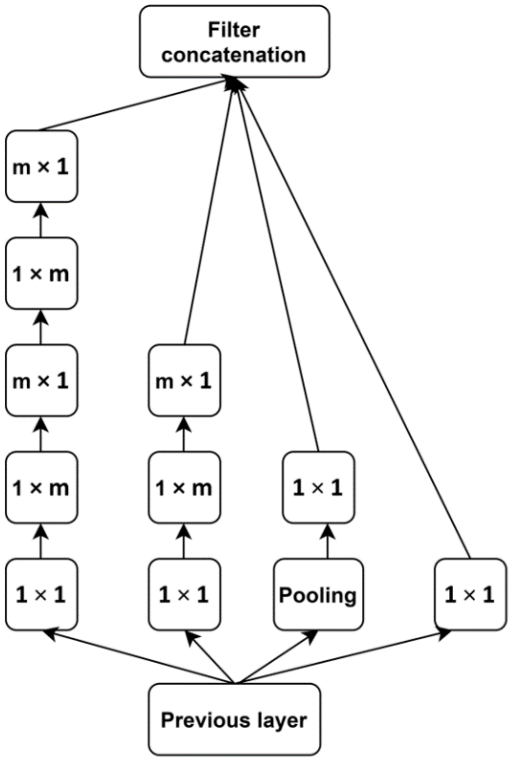


Figure 5.5 : Inception module after factorization of m×m convolutions (Szegedy et al., 2016).

The inception module is illustrated in Figure 5.5; when m equals three, this inception module matches the previous module illustrated in Figure 5.4. The left 5×5 convolution is represented as 1×3, 3×1, 1×3, and 3×1 in series.

The authors proposed another inception module where filter banks are made wider instead of deeper to avoid the representational bottleneck problem. Information loss can occur when dimensions of the input decrease excessively because of a deeper module. The inception module is made wider, as illustrated in Figure 5.6.

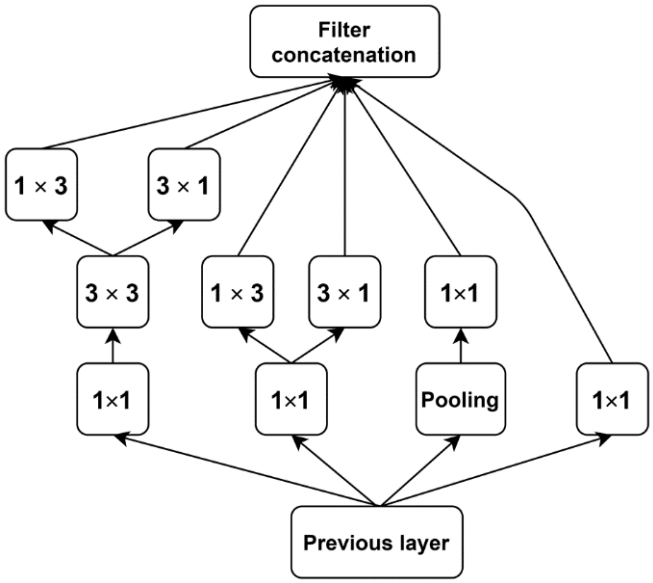


Figure 5.6 : Building a wider inception module (Szegedy et al., 2016).

InceptionV2 architecture uses the three inception modules illustrated in Figure 5.4, Figure 5.5, and Figure 5.6, besides other convolution, pooling, and classification layers. The architecture is given in Table 5.2, which presents layers and input sizes for each of these layers.

InceptionV3 includes all of the improvements of InceptionV2 that are mentioned above and has additional improvements. InceptionV3 uses the root mean squared propagation (RMSProp) algorithm as an optimizer. Batch normalization is used in auxiliary classifiers. Besides convolutions, batch normalization is also used in the FCL of auxiliary classifier.

Table 5.2 : InceptionV2 model layers and input sizes.

layers	input size
convolution	299×299×3
convolution	149×149×32
convolution padded	147×147×32
pooling	147×147×64
convolution	73×73×64
convolution	71×71×80
convolution	35×35×192
3×inception module in Figure 5.4	35×35×288
5×inception module in Figure 5.5	17×17×768
2×inception module in Figure 5.6	8×8×1280
pooling	8×8×2048
linear	1×1×2048
softmax	1×1×1000

Auxiliary classifiers term is introduced by Szegedy et al. (2015) to help the convergence of deep models. The motivation for using the auxiliary classifiers is to push backward beneficial gradients to the lower layers to make these gradients instantly useful and enhance the convergence along the training process by reducing the impact of the vanishing gradient problem in deep networks. Szegedy et al. (2015) used auxiliary classifiers at different stages. The network's performance was not reduced when the lower auxiliary branch was removed.

Auxiliary classifiers didn’t improve convergence at the beginning of training, but near completion of training, the network with auxiliary classifiers begins to pass the accuracy performance of the network that doesn’t have an auxiliary classifier. Auxiliary classifiers act like a regularizer. Performance of the main classifier increases if the auxiliary classifier uses a dropout layer, or is batch normalized (Szegedy et al., 2016). The black box in Figure 5.7 illustrates the auxiliary classifier of the InceptionV3.

Factorized 7×7 convolutions are applied where the first 7×7 convolution layer is factorized into a series of 3×3 convolutional layers.

Label smoothing as a regularization component is applied (added loss function) to prevent the model turns out overconfident. This regularization also helps to overcome overfitting. Cross entropy is computed with a weighted mix of targets from the dataset with uniform distribution instead of challenging targets (Szegedy et al., 2016).

InceptionV3 has a depth of 159 layers and 23,851,784 parameters when input image size 299×299 in Keras applications. The model uses building blocks that include convolution layers, inception modules, pooling (average and max), concatenation, dropouts, batch normalization, and fully connected layers.

The architecture of InceptionV3 is illustrated in Figure 5.7. The input images in the shape of 299×299×3 are fed to the model, and outputs in size of 8×8×2048 are obtained.

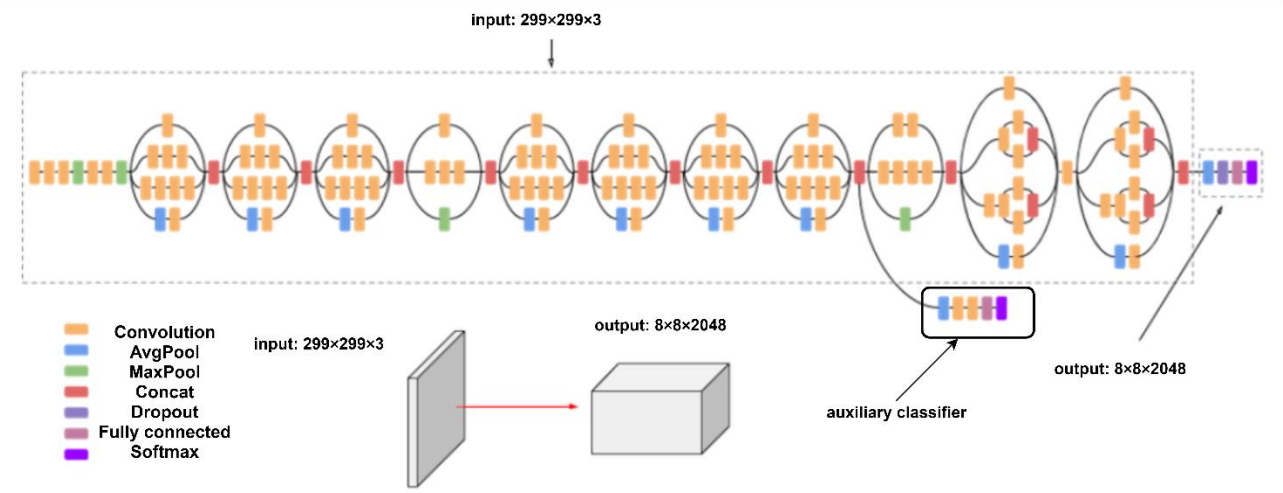


Figure 5.7 : Architecture of InceptionV3 (URL-9).

5.5 Feature Selection

Feature selection has become popular because of its applications in many areas, such as image processing, computer vision, text mining, bioinformatics, etc. It is essential in data processing and is one of machine learning research's most challenging and critical problems (Çakmak et al., 2021). The large data size to be processed causes problems such as prolonged training and over-compliance (Kumari et al., 2021).

A dataset can include a large number of features/attributes. Sometimes all of the features are not necessary to obtain beneficial information in the dataset. Because some features can be redundant, unnecessary, or irrelevant, which reasons to decrease in model performance. So decreasing the number of features while maintaining the accuracy performance of the model is the primary purpose of feature reduction. Feature selection and feature construction or extraction fall within feature reduction methods. The difference between these two methods is as follows: The feature construction or extraction constructs a new feature (brand new ones) set from a dataset.

On the other hand, feature selection selects related features from the dataset. A subset of original features is kept (Agrawal et al., 2021). Feature selection algorithms perform well when features are irrelevant or considerably correlated with class labels (Tahir et al., 2007).

In general, feature selection methods are divided into three classes: filter, embedded, and wrapper methods, illustrated in Figure 5.8. These methods will be described shortly (Liu et al., 2010; Liu and Zhao, 2012).

Filter methods concentrate on the overall characteristics of data. They are independent of classification or learning algorithms. Features are selected depending on their relationship with the target. Statistical techniques are used to assess the relationship between target and input variables; then, features are chosen (filter) based on this statistical information. On the other hand, many models are generated, which include different feature subsets in wrapper methods. These methods have a classification/learning algorithm in the evaluation step of the feature subset. A classification algorithm evaluates the goodness of the selected features.

When wrapper methods are compared with filter methods, it is seen that they have more computational expense than filter methods, while their accuracy performance is more than filter methods'. Embedded methods are an integration of wrapper and filter methods. (Xue et al., 2015; Agrawal et al. 2021).

Wrapper methods use a modeling algorithm where each subset is formed and interpreted. Subset formation resides in different search strategies. The search strategies can be divided into three classes: sequential, randomized, and exponential strategies. An exponential strategy is computationally expensive because when the number of features increases, the number of interpreted features increases exponentially. Exhaustive search is an example of an exponential search strategy. The sequential strategy involves or takes out features sequentially. When it is decided to involve or take out a feature in the selected subset, it cannot be changed anymore, leading to local optima.

Randomized algorithms, known as population-based approaches, use randomness when exploring search space. Randomness helps algorithms to avoid local optima. Metaheuristic algorithms are randomized algorithms (Agrawal et al., 2021).

In this study, three metaheuristic algorithms are applied in feature selection.

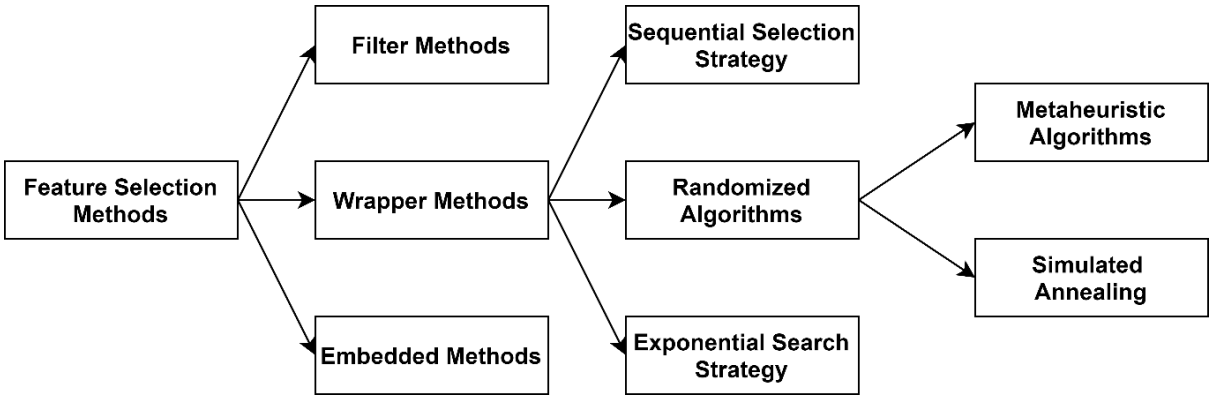


Figure 5.8 : Feature selection methods (Agrawal et al., 2021).

5.6 Stochastic Optimization

An optimization process can be defined in general as follow: inputs are fed into a function, and according to the goal of the objective function, minimization or maximization is done.

Stochastic is synonymous with probabilistic and random. In stochastic optimization, randomness is used as a search process strategy in the optimization algorithm or objective function. Due to randomness usage, any single run of a stochastic algorithm will result in different starting points, decisions, etc., along the search process. Randomness can prevent the local optima trap and increase the chance of reaching the global optimum (Kochenderfer and Wheeler, 2019). Several machine learning algorithms are stochastic because these algorithms apply randomness in optimization. Such as stochastic gradient descent or stochastic gradient boosting.

Stochastic optimization algorithms are applied as an alternative to deterministic algorithms. These algorithms enable overcoming given system noise and highly dimensional or highly nonlinear problems unfavorable to classical deterministic algorithms (Spall, 2012).

A feature selection problem can be considered an optimization problem. All combinations of input features can be utilized during the selection process, and the optimum subset is found. When feature number is increased dramatically, stochastic optimization techniques can promise to obtain a successful subset of features.

5.7 Metaheuristic Algorithms

Developments in artificial intelligence research have caused the development of intelligent optimization algorithms that provide near-optimal solutions to complex and challenging optimization problems in the real world. Moreover, these recent developments have also led to developments in metaheuristic optimization algorithms (Ezugwu et al., 2021).

Heuristic and metaheuristic are stochastic methods. Heuristic and metaheuristics are often used interchangeably. The difference between them is inappreciable. Both algorithms are applied to find satisfying solutions quickly, where it is difficult to find an optimal solution. A heuristic has a reasoning methodology that solves problems by trial and error. In general, heuristic algorithms are problem dependent without generalization possibility, while metaheuristic algorithms are problem-independent techniques. A metaheuristic is a higher-level heuristic. Metaheuristic algorithms have adaptive computing and use heuristic rules during problem-solving (Stojanović et al., 2017; Hussain et al., 2019; Ezugwu et al., 2021). One of the shortcomings of heuristic algorithms is to be trapped in a local optimal solution. The metaheuristic algorithms have been developed to cope with this shortcoming (Rajabi Moshtaghi et al., 2021).

Metaheuristic algorithms are a kind of approximate algorithms and are usually non-deterministic. These algorithms are applied widely because of their easy implementation, simplicity, skill in avoiding local optima, being not specific to problems, and they accomplish not differentiable problems. Metaheuristics are successful in problems where mathematical (exact or traditional) optimization is impractical because of imperfect problem definitions and a long time for computation. Metaheuristic algorithms use techniques that range from easy local search methods to complicated learning processes. Metaheuristic has two essential characteristics: exploitation and exploration. Exploration is related to the skill of algorithms to explore new search areas, while exploitation tries to find the best solution in a promising area of search space. The balance between exploitation and exploration is critical (Blum and Roli, 2003; Mohamed et al., 2020; Stegherr et al., 2020).

The terms diversification and intensification are also used. Diversification is related to search space exploration, whereas intensification is related to the exploitation of cumulative search experience. The terms exploitation and exploration mainly address

short-term strategies used with randomness, while intensification and diversification address medium- and long-term strategies with memory usage (Blum and Roli, 2003).

A metaheuristic algorithm cannot perform equally well on all optimization problems. This and other reasons have led to the development many nature-inspired metaheuristic algorithms over the past years. On the other hand, many recent algorithms are based on a metaphor of some man-made or natural process. These metaphors are increasingly exaggerated, unrelated to optimization in nature. New algorithms lack improvements and are similar to existing algorithms. Additionally, there are problems with these algorithms' testing and statistical analysis. Researchers should concentrate on more promising research directions (Sörensen, 2015; Stegherr et al., 2020).

In the literature, there are different categories of metaheuristic algorithms. Classification of nature-inspired versus non-nature-inspired metaheuristics is based on the origins of algorithms. Because of hybridization usage in recent algorithms or failure in assigning algorithms to a class clearly, this classification is not always strictly valid (Blum and Roli, 2003). The natural processes have simply classified biological processes versus physical laws-based algorithms (Ezugwu et al., 2021).

Global search optimization algorithms are more explorative than local search algorithms, while local search algorithms are generally exploitative methods (Hussain et al., 2019). Single-point search versus population-based classification is related to the number of solutions the algorithm handles at any time. Algorithms with static objective function do not modify objective function during the search process. On the other hand, algorithms with dynamic objective function update objective function to avoid local minimum. Another classification is one versus various neighborhood classification. This classification is interested in changing the fitness landscape during the search or not changing. Memory usage is another important subject in classification. Memory-less algorithms apply the Markov process. The algorithms with short-term memory care recent decisions more. An aggregation of synthetic parameters is used in search of long-term memory algorithms (Blum and Roli, 2003).

The classes of metaheuristic algorithms inspired by nature are given in Brownlee (2011). These classes are genetic algorithms, swarm optimization algorithms, science-based algorithms such as physics, chemistry, or social science-based, optimization

algorithms with multi/many objectives, algorithms of evolutionary computation, physical algorithms, and immune algorithms.

Metaheuristic algorithms are applied in a wide range of optimization problems in engineering, mathematics, finance, management/industry, natural science, earth science, social, biological sciences, etc. Some real-world applications of metaheuristics are presented in Figure 5.9 (Soler-Dominguez et al., 2017; Ezugwu et al., 2021).

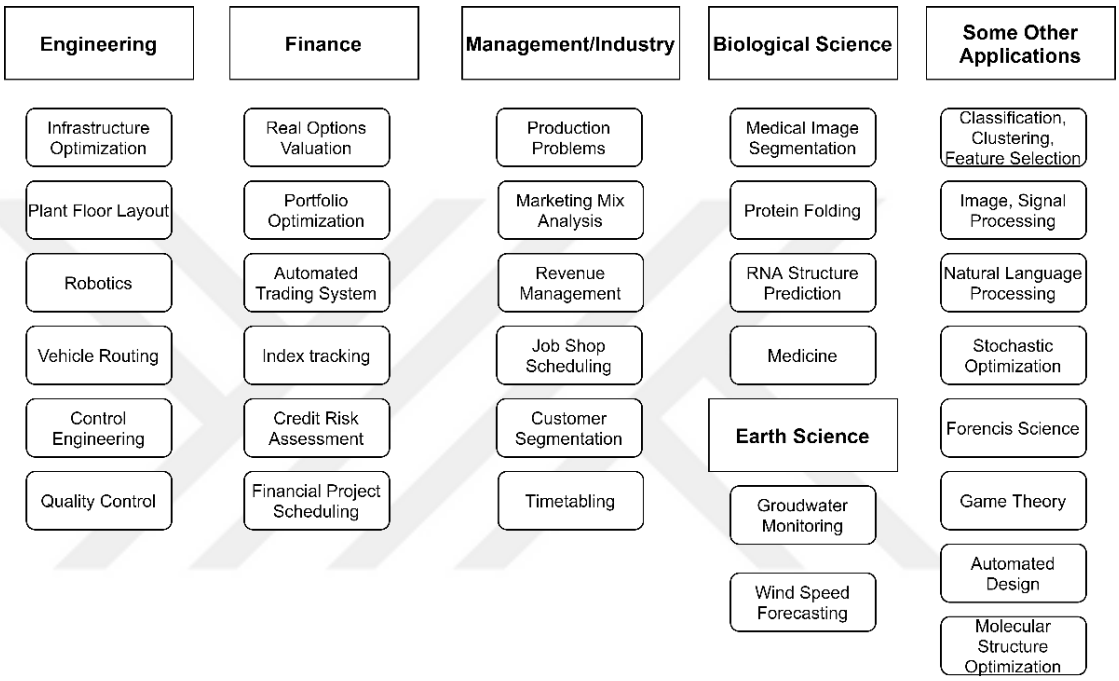


Figure 5.9 : Some application areas of metaheuristics.

Metaheuristic algorithms have the capability of getting the best feature subset as well as maintaining the accuracy performance of a model. These algorithms promise to solve several kinds of optimization problems and are applied to feature selection problems (Agrawal et al., 2021).

Wrapper strategy-based metaheuristic algorithms can select optimal feature subsets and tune a classifier's parameters, which in turn increases classification accuracy (Wang et al., 2014).

5.7.1 Classifier selection for metaheuristic algorithms

Classifier selection for developing an algorithm of wrapper feature selection is critical to evaluating the goodness of selected features (Xue et al., 2015). Different classifiers are used in feature selection with metaheuristic algorithms, such as SVMs, Naïve

Bayesian, Artificial Neural Networks, Random Forest, k-Nearest Neighbors (kNN), and Fuzzy Rule-Based. A literature survey by Agrawal et al. (2021) presents that the most frequently used classifier in feature selection problems (in UCI datasets) is kNN, followed by SVM. The most common use of these classifiers is their high performance and ease of use. Furthermore, kNN is more advantageous in large dimensional datasets. Such as Tahir et al. (2007), Gunavathi and Premalatha (2015), Darwish et al. (2018), and Canayaz (2021) applied the kNN classifier in the feature selection fitness function.

5.7.1.1 k- nearest neighbors algorithm

The k-nearest neighbors (kNN) algorithm is a non-parametric and instance-based classifier method proposed by Fix and Hodges (1951) firstly, later developed by Altman (1992). It is used in both regression and classification problems.

kNN supposes that data points are found in a feature or metric space, and there is a notion of distance between these points. In a classification problem, each training data includes a set of vectors and related class labels. An integer value "k" is used ($k \geq 1$), which decides how many number of nearest neighbors (based on a distance metric) affects classification.

During the kNN classification process, k number of nearest neighbors of an unknown instance is chosen from the training set. Then, class label prediction is made by looking at the most frequent label occurring in selected neighbors (Tahir et al., 2007). The neighbors are selected using some similarity or distance measure. Such as Euclidean, Manhattan, Jaccard, Minkowski Distance, or Cosine Similarity. In the basic application of kNN, uniform weights are used, where the classifier assigns a query point into a class using the majority vote of nearest neighbors. In some cases, weighting the neighbors according to their contribution is better. A frequently used weighting is assigning weights in direct proportion to the inverse of the distance from the related query point (URL-10).

When the degree of local sensitivity increases, the kNN classifier becomes more vulnerable to noise in training data (Gunavathi and Premalatha, 2015). Additionally, kNN is vulnerable to the high dimensionality of data. kNN performs better when data is on the same scale; data can be normalized or standardized. The missing data problem should be solved by excluding related samples or imputing missing values.

The value of k depends on the dataset. In binary classification tasks, k is generally chosen as an odd number, such as 3, 5, 7, etc. Generally, a high value of k decreases the effect of noise on the classification. However, classification boundaries turn out less distinct (Puspadini et al., 2020). Increasing k also increases the computation cost.

In this study, the vector of extracted features consists of real values for each image. With feature selection algorithms, another feature vector is obtained. This vector consists of 0 or 1 string value for each feature. 0 means that the feature is not selected, and 1 means the feature is selected in the subset.

kNN is applied as a classifier in evaluating each solution (feature set). The k value is chosen by trial and error, and Euclidean distance is used in neighbor calculation; the formula is given in equation 5.1. Euclidean distance measures the distance from two points in two or more dimensions.

Let a1 and a2 be two different samples in the dataset. a1 is denoted by $a_1 = (a_{11}, a_{12}, \ldots\ldots, a_{1n})$ and a2 is denoted by $a_2 = (a_{21}, a_{22}, \ldots\ldots, a_{2n})$. n is the dimension of Euclidean space.

$$D\,(a1, a2) = \sqrt{\sum_{i=1}^{n} (a_{1i} - a_{2i})^2} \qquad (5.1)$$

For the fitness function, which is given in explanations of metaheuristic algorithms, cross-validated classification loss (kfoldloss) is used.

Classification loss was obtained from the cross-validated classification model, kNN. For every fold, this method calculates classification loss for in-fold (validation fold) observations using a model (kNN) trained on out-of-fold (training fold) observations.

As the loss function of kfoldloss, minimal expected misclassification cost (mincost) is used. By default, kNN models give posterior probabilities as classification scores. Mincost is suitable for this classification score.

In this study, the software used in the Matlab library calculates the weighted minimal expected classification cost applies the following procedure (URL-11):

For observations i = (1, 2, 3, ….., m), the expected misclassification cost for classifying observation X_i into the class m' is estimated using equation 5.2, where f (X_i) returns

class posterior probabilities of the classification for observation X_i. C is the cost matrix.

$$\gamma_{im} = (f(X_i)'C)m, \tag{5.2}$$

For observation i, class label prediction is made the corresponding mincost using equation (5.3).

$$\widehat{y_i} = \underset{m=1,2,\ldots,m}{\operatorname{argmin}} \gamma_{im'} \tag{5.3}$$

The cost incurred (c_i) is identified using C for the prediction. The weighted average of mincost loss is calculated using equation (5.4). Cost value is 1 for incorrect classification and 0 for correct classification.

$$L = \sum_{i=1}^{m} w_i c_{i'} \tag{5.4}$$

5.7.2 Particle Swarm Optimization

Particle swarm optimization (PSO) is a population-based stochastic algorithm. It is inspired by fish and bird behaviors. PSO optimizes a given problem iteratively by trying to improve a candidate solution regarding a given quality measure (Sarı et al., 2021). It differs from other algorithms in that way; PSO doesn't reside in the gradient of the objective function.

Binary PSO was used for feature selection, the binary version of PSO (Jiang et al., 2017). The particle's velocity in the swarm is updated using equation 5.5, and the position of the particle is changed by using equations 5.6 and 5.7. Each particle in the swarm represents a possible solution. The possible solution is a 0/1 (1x feature number) matrix, and each column value is randomly generated. A possible solution (a particle) is obtained by selecting the column numbers with a 1 in the feature matrix. kNN classifier error rate was used for the fitness value of particles in the swarm. In the first iteration, the particle with the swarm's best objective function value is assigned as gbest. In the subsequent iterations, the particle with the best objective function value is assigned (5.8). If pbest has a better objective function value than gbest, then pbest is assigned as gbest (5.9). This process is continued through T iteration, and the algorithm provides the best solution globally as output (Canayaz, 2021).

$$v_i^d(t+1) = wv_i^d(t) + c_1 r_1 \left(pbest_i^d(t) - x_i^d(t)\right) + c_2 r_2 \left(pbest^d(t) - x_i^d(t)\right) \tag{5.5}$$

$$S\left(v_i^d(t+1)\right) = \frac{1}{1+\exp\left(-v_i^d(t+1)\right)} \tag{5.6}$$

$$x_i^d(t+1) = \begin{cases} 1, & \text{if rand} < S(v_i^d(t+1)) \\ 0, & \text{otherwise} \end{cases} \tag{5.7}$$

$$pbest_i^d(t+1) = \begin{cases} x_i(t+1), & \text{if } F\left(x_i(t+1)\right) < F\left(pbest_i(t)\right) \\ pbest_i(t), & \text{otherwise} \end{cases} \tag{5.8}$$

$$gbest(t+1) = \begin{cases} pbest_i(t+1), & \text{if } F\left(pbest_i(t+1)\right) < F\left(gbest(t)\right) \\ gbest(t), & \text{otherwise} \end{cases} \tag{5.9}$$

The pseudo-code of the feature selection with PSO is given in Table 5.3.

Table 5.3 : Pseudo-code of feature selection with PSO.

Set parameters N=20, T=100, kfold, features
Generate the initial swarm randomly For i=1 to N Generate 0/1 matrix size (features,2) Select feature if column values=1 End Calculate the objective function for each particle in swarm by using kNN classifier Find Best Particle in Swarm $gbest$=min of Swarm While $t < T$ Update velocity $v_i^d(t)$ by using equation 5.5 Update position $x_i^d(t)$ by using equations 5.6 and 5.7 For i=1 to N Generate 0/1 matrix size (features,2) Select feature if column values =1 Find Best Particle in Swarm $pbest$=min of Swarm End Calculate the objective function for each particle in swarm by using kNN classifier If $gbest > pbest$, Then $gbest = pbest$ End while Output $gbest$

5.7.3 Simulated Annealing

Simulated Annealing (SimAn) is one of the popular heuristic methods widely used to solve combinatorial optimization problems. The advantage of this approach is that it does not stick to the local optimum (Wu et al., 2011). Random numbers with a uniform distribution between 0 and 1 were generated for the initial solution (S). If the randomly

generated number is greater than 0.5, the relevant feature is included in the solution. The objective function value (S_{OBJ}) of the initial solution is calculated using the kNN classifier. Neighbor solution (N) is created similar to the initial solution. The objective function value (N_{OBJ}) of the neighboring solution is calculated. If the neighbor solution objective function (N_{OBJ}) is better than the solution objective function value (5.10), the solution is replaced by the neighbor solution (S=N). If the neighbor solution does not improve the objective function's value, the solution is not changed with the probability of $e^{-\Delta Obj/T}$, and the next iteration is passed. The annealing temperature is calculated using the geometric ratio. The annealing temperature (5.11) is updated by multiplying the annealing (T) with the cooling coefficient (C_r=0.999) at each iteration. The iteration process is continued until $T=T_{end}$ (Kim et al., 2002).

$$\Delta Obj = N_{OBJ} - S_{OBJ},$$
$$\Delta Obj < 0 \tag{5.10}$$

$$T = T * Cr \tag{5.11}$$

The pseudo-code of the feature selection with SA is given in Table 5.4.

Table 5.4 : Pseudo-code of feature selection with SA.

Set parameters features(i), T, Cr, T_{end}

Generate initial Solution
- d← Dimension of features
- Repeat
- if rand(1,i) > 0.5 select feature
- Until i=d

Calculate Obj by using kNN

Repeat
- Generate Neighbor Solution
- Repeat
- if rand(1,i) > 0.5 select features(i)
- Until i=d
- Calculate *NObj* by using kNN
- Calculate *ΔOBJ* ← *NObj-Obj*
- if *ΔOBJ<0,* then *S←N*
- if *ΔC<0* or $e^{-\Delta C/T}$>*rand(0,1)* then
- S←N
- T←T*Cr
- Until $T=T_{end}$

Output *S*

5.7.4 Artificial Bee Colony

The artificial bee colony (ABC) algorithm is proposed firstly by Karaboga (2005). The ABC algorithm is inspired by the foraging behaviors of honey bees and is a population-based optimization algorithm. The algorithm procedure is summarized in the general sense as follows: Bees fly around, and the aim of a bee is to discover nectar resources (food) and try to find the place with the most nectar finally. Bees, in the progress of time, update the food resource positions. There is a division of labor between bees. There are three groups of bees: onlooker, employed, and scout. Onlooker and employed bees decide on food resources based on their experience and nest mates. Employed bees use previous information from resources, and this information is shared with onlooker bees. According to this information, bees arrange their positions. Scout bees fly around and decide on food resources randomly, in other words, without using experience. If the amount of nectar found in a new resource is more than the previous one, the new position is memorized, and the previous position is forgotten. So ABC algorithm uses both local and global search methods (URL-12).

The basic procedure of the ABC algorithm has four steps based on (Lin et al., 2021; Uzer et al., 2013).

The first step is initialization. Each resource of food states a possible solution that is presented as a vector: A = ($a_{i,1}$, $a_{i,2}$, $a_{i,3}$, …., $a_{i,d}$), and is formed using equation (5.12). (x), where i = (1, 2, 3, ….., FSN) and FSN stands for the number of food resources. j = (1, 2, 3, ….., d), here d stands for search space dimensionality.

$$a_{i,j} = a_j^{min} + \text{rand}\,(0,1)\left(a_j^{max} - a_j^{min}\right) \tag{5.12}$$

The second step is the process of the employed bee. Each employed bee is related to a food resource. Employed bees need to change the food resource position to find better positions. So, these bees look for neighbor resources that are generated randomly. Equation 5.14 presents producing the new resource (a candidate resource position). The position of a resource and resource amount in a resource depends on the quality (fitness) of the associated solution, which is calculated with equation (5.13).

$$\text{fitness}_i = \frac{1}{1 + f_i} \tag{5.13}$$

In equation 5.14, $\beta_{i,j}$ is a random value, distributed uniformly and is ranged between [-1, 1]. $\beta_{i,j}$ enables control of the generation of neighboring resources around a_{ij}. n ϵ {1,

2, 3, ….., FSN}. n and j are selected randomly. After $a_i^{'}$ is generated, its fitness value is evaluated. kNN error rate is used as f_i. Then a selection is made by comparing $f(a_i^{'})$ and $f(a_j)$ values. Accordingly, it becomes clear which one will enter the next iteration.

$$a_{i,j}^{'} = a_{i,j} + \beta_{i,j}\left(a_{i,j} - a_{n,j}\right) \tag{5.14}$$

The third step is the process of the onlooker bee. Employed bees share resource positions and nectar amount (fitness value) information with onlooker bees, and onlooker bees decide a food resource considering fitness proportionate selection (roulette wheel scheme). A better fitness value of a source increases the probability of being selected. The probability for each resource is calculated in equation (5.15).

$$P_i = \frac{\text{fitness}_i}{\sum_{j=1}^{FSN} fitness_j} \tag{5.15}$$

A random number between (0,1) is generated to decide the choice of food resource. If P_i is greater than generated random value, a_i is selected and updated like in the process of the employed bee.

The last step is the process of the scout bee. Each resource has a counter value; it is zero at the beginning of the algorithm. When the counter holding value exceeds the abandonment limit value (AL), that means its related food resource will be abandoned, and a new resource will be replaced. It is obtained using equation (5.12).

The basic procedure of the ABC algorithm is proper for optimization problems in continuous space, but future selection problem is in discrete space.

So, the value found from the formula in equation 5.12 should be turned into a discrete value using equation (5.16). Each feature subset is expressed in a binary way. According to equation 5.16, if the value of a dimension is less than 0.5 (threshold value), the related feature is not selected, and then its value becomes “0”. Otherwise, it is selected, and its value becomes “1”.

$$a_{i,j} = \begin{cases} 0, & \text{if } a_{i,j} < 0.5 \\ 1, & \text{otherwise} \end{cases} \tag{5.16}$$

The pseudo-code of the feature selection with ABC is given in Table 5.5.

Table 5.5 : Pseudo-code of feature selection with ABC (Lin et al., 2021).

```
Set parameters FSN, MaxIter,  AL, t=0, counter=0
Initialize population a_i, (i = 1,2,3,…., FSN) with using 5.12
Evaluate fitness value of each food resource
While t ≤ MaxIter do
    %Employed bee process
    for each employed be do
        Choose a different food resource a_n randomly
        Generate a new food resource using 5.14 and convert to discrete
        values using 5.16
        Interpret fitness value of each food resource
        Update a_i in terms of greedy selection and enhance its counter by
        1
        If not update
    end for
    Calculate the selection probability of each food resource using 5.15
    %Onlooker bee process
    for each onlooker be do
        Choose a food resource a_i in terms of selection probability by
        fitness proportionate selection
        Choose a different food resource a_n randomly
        Generate a new food resource using 5.14 and convert to discrete
        values using 5.16
        Interpret fitness value of each food resource
        Update a_i in terms of greedy selection and enhance its counter by
        1
        If not update
    end for
    %Scout bee process
    for each food resource, do
        if counter ≥ AL, then
                Replaced by a new food resource using 5.14 and convert
                to discrete values using 5.16
                Interpret fitness value of new food resource
        end if
    end for
end while
Output a_best, f(a_best)
```

5.8 Classification

In this study, extracted features are classified with various machine learning methods. Some of the classifiers are based on ensemble learning. These classifiers are Extreme Gradient Boosting (XGBoost), Bagged Decision Trees, Extra Trees, and Random Forest classifier. The other classifiers are Logistic Regression, SVMs, and MLP. Detailed computational explanations of these classifiers are beyond the scope of this

study. A summary of introductory information or the generalized algorithm is given for each classifier.

5.8.1 Ensemble methods

Recently, multiple classifiers or ensemble systems have attracted attention in wide application areas such as confidence estimation, feature selection, error correction, and incremental learning problems in machine learning. Ensemble methods are primarily developed to decrease the variance, thus increasing models' accuracy performance in decision-making systems (Polikar, 2012).

Three cases pioneered developments in current ensemble methods. These cases are combining of classifiers, the ensemble of base/weak learners, and a mix of experts.

The combining of classifiers method was commonly applied in pattern recognition research. The researchers use robust classifiers and try to build effective combining rules to obtain superior combined classifiers in general. There is a need for a deep understanding of building and using different combining rules in the combining classifiers method. Ensemble of base learners was commonly applied in the machine learning area. In this case, researchers frequently design the models using weak learners and aim to boost the weak learners to strong ones. Lastly, the mix of experts was commonly applied in the area of the neural network. In this case, researchers try to collectively learn a mix of parametric models and apply combining rules to obtain a comprehensive solution (Zhou, 2019).

Ensemble methods use multiple learners to solve the same problem. The main idea of ensemble methodology is to train multiple pattern classifiers and combine them to attain a classifier that surpasses them (Rokach, 2019). Ensemble learning is called committee-based learning or learning multiple classifier systems.

A basic ensemble learning architecture is presented in Figure 5.10. An ensemble model includes several learners that are called base learners. These base learners are generally formed from a training dataset by a learning algorithm such as a decision tree, SVMs, and neural networks. Many ensemble models use the same type of base learner, so called homogeneous ensembles. On the other hand, when multiple algorithms are used as base learners, it is called heterogeneous ensembles. Some researchers call base learners individual or component learners in heterogeneous ensembles (Zhou, 2019).

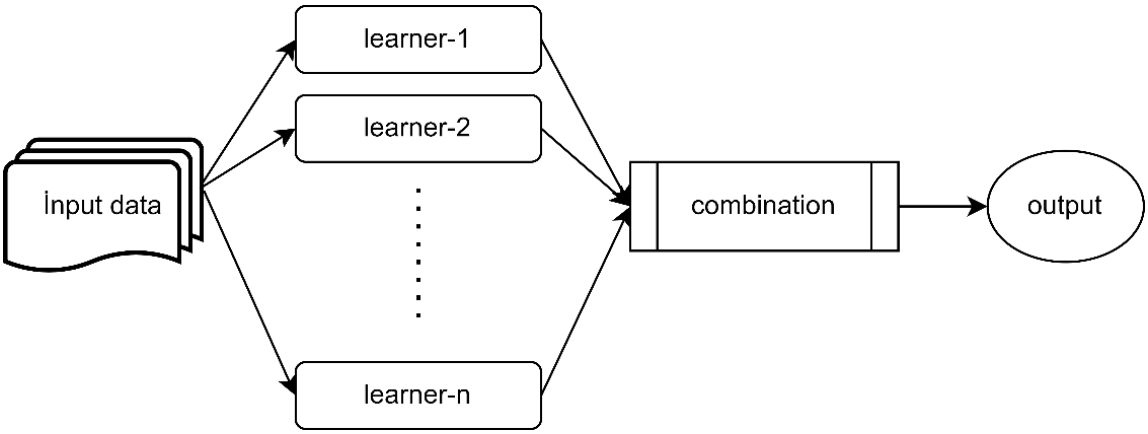


Figure 5.10 : A basic ensemble learning architecture.

In terms of generalization ability, an ensemble is superior primarily in performance than base learners (weak learners). Ensemble methods can boost or promote weak learners, giving very few better performances than random predictions to strong learners that can predict very accurately (Zhou, 2019). On the other hand, an ensemble model doesn't guarantee to give better classification performance than the best classifier of the ensemble model. Still, it decreases the chance of selecting a weak classifier (Polikar, 2012).

The classification error is comprised of two ingredients that can be controlled. These are bias and variance. They are related to the accuracy and precision of the classifier on various training datasets. The bias and variance ingredients often have a tradeoff relation. A classifier with a low variance tends to have a high bias and vice versa. Additionally, it is known that averaging offers a variance-reducing effect (Polikar, 2012).

According to recent studies in the machine learning field, combining the outputs from several classifiers decreases the generalization error of individual classifiers. The success of ensemble methods is mainly based on biases; namely, different classifiers have different inductive biases. These methods can use such diversity effectively to decrease the variance error without an increase in bias error (Rokach, 2019).

Ensemble models differ from each other related to combining rules (such as algebraic rules) for ensemble decisions, the method applied in ensemble members generation, and the selection of training data for each classifier (Polikar, 2012). The mainly used ensemble techniques are bootstrap aggregating (bagging), boosting, and stacking in machine learning.

In the bagging technique, a single machine learning algorithm is typically used. This algorithm is nearly always an unpruned decision tree. Each base learner is trained on a different sample of the training dataset. Bootstrap sampling is made to obtain data samples; instances are drawn from the training dataset randomly and with replacement. Then, the predictions of base learners (ensemble members) are combined using some statistics such as averaging or voting (URL-13). A bagging architecture is given in Figure 5.11.

More details about the bagging classifier, random forest classifier, and extra trees classifier are explained in the following sections.

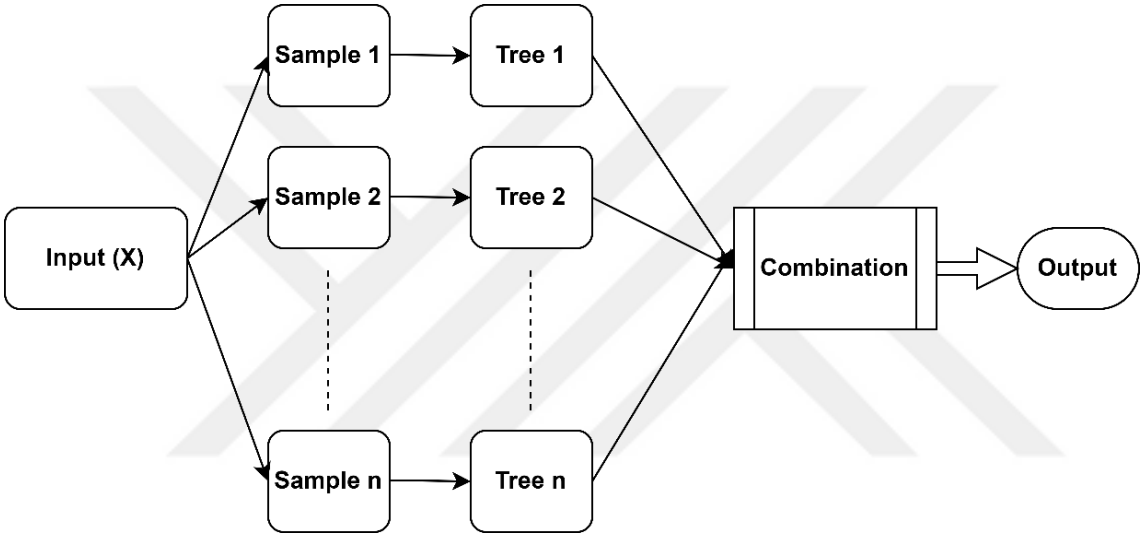


Figure 5.11 : Bagging architecture.

The bagging technique benefits from the instability which is inherent in learning algorithms. On a hunch, combining several models is helpful in the case these models are significantly different from each other, and each model is trained with an acceptable percentage of data properly. Complementing of models one another is sought. In the boosting technique, several models are combined by seeking complementary models. A boosting model uses some combination rule (such as voting, averaging, etc.) to obtain the output from individual models, similar to bagging (Witten et al., 2016).

Additionally, it combines the same types of models, such as decision trees like bagging. On the other hand, a boosting model is iterative: base learners in bagging are built separately. In boosting, each new model is affected by the performance of previously added models. Boosting promotes new models to become experts, for

instances misclassified by previously generated models. The models are fitted and added to boosting model sequentially. The second model tries to correct the predictions made by the first model; the third model tries to correct the predictions made by the second model, and so on. Predictions are combined by a weighted average of models. In other words, the contribution of a model is weighted according to its performance rather than assuming an equal weight for all models (Witten et al., 2016). A boosting architecture is given in Figure 5.12.

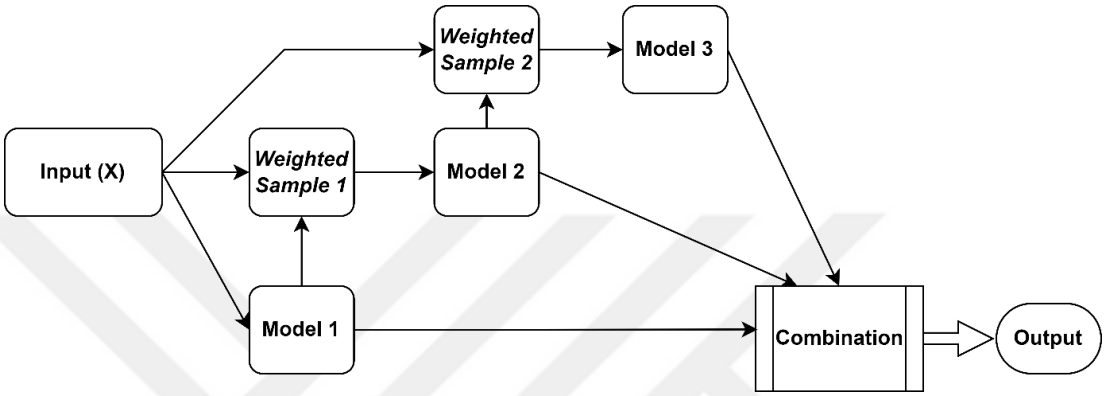


Figure 5.12 : Boosting architecture.

Stacking is an overall model that is a generalization of several ensemble methods. It is a particular combination technique that combines by learning (Zhou, 2019). The stacking technique combines different prediction models in a single model. A stacking model has layers/levels in the concept of meta-learning, where machine learning algorithms learn from the output of other algorithms (Ribeiro and dos Santos Coelho, 2020).

A stacking model aims to learn which classifiers are robust using another learning algorithm called meta learner. How to best combine the predictions from base learners is learned (Witten et al., 2016). Let's assume a stacking model has two levels/layers: level 0 and level 1. The number of levels can be more. In level 0, different machine learning algorithms are trained, and predictions are obtained. Then, these predictions are used as input for an algorithm in level 1. This algorithm in level 1 is trained, and the result is obtained (URL-14).

In summary, the algorithm in level 1 learns with predictions made by algorithms of level 0. Diversity among algorithms of different levels can provide advantages in the results. It is preferable to obtain prediction errors that are less correlated with this

diversity (Ribeiro and dos Santos Coelho, 2020). A stacking architecture is given in Figure 5.13.

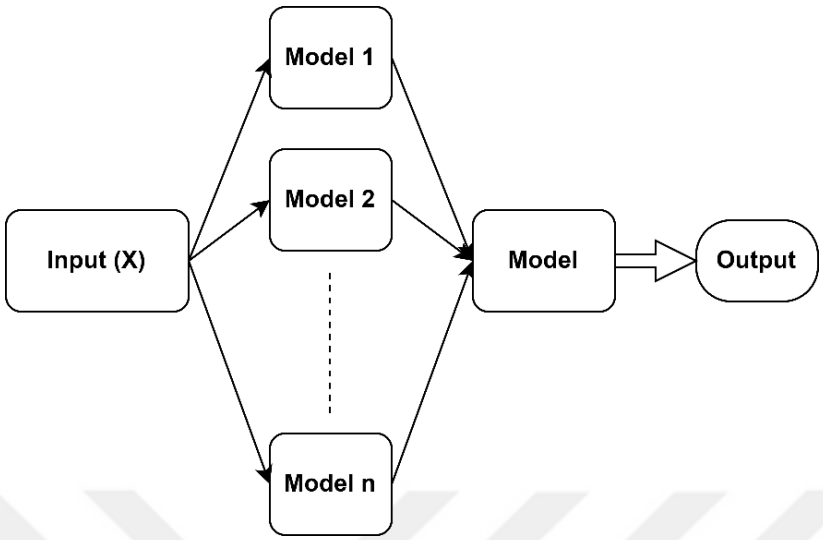


Figure 5.13 : Stacking architecture.

During the training of a stacking model, there is a risk of overfitting when exact data used in the train first-level models are also used to form new data sets for training second-level models. The cross-validation or leave one out technique is suggested to prevent overfitting (Zhou, 2019).

5.8.2 Extreme gradient boosting

Tree boosting methods are very effective and widely applied in many machine learning challenges. XGBoost is short for “Extreme Gradient Boosting” and is a scalable end-to-end tree boosting system (Chen and Guestrin, 2016). It is a supervised learning machine learning system that can handle tabular or structured datasets in regression and classification tasks.

XGBoost is developed based on a gradient boosting decision tree algorithm. This algorithm is called by a different name, for example, multiple additive regression tress, gradient boosting machines, stochastic gradient boosting, or the most used one, gradient boosting.

Boosting is a kind of ensemble model in which new models are included to correct errors made by present models. Models are attached in order until no further progress is made.

Gradient boosting is a technique in which new models are generated that predict errors or residuals made by previous models and later added jointly to perform final prediction. Its name comes from the gradient descent algorithm. When new models are added, the gradient descent algorithm minimizes loss (Brownlee, 2016).

Similar to other boosting algorithms, there is a dependence between the training of each model and the models already trained in gradient boosting. The learning process aims to build the base models correlated in maximum with the negative gradient of loss function related to the whole ensemble. In other words, regression models in a sequence are calculated, in which each successive model predicts pseudo residuals of antecedent models given a differentiable loss function. This loss function is needed to calculate the negative gradient. The loss function is minimized during the aggregation of predictions. The gradient boosting models generally have many simple models in response to other ensemble methods with less but more complicated models. During the training, deciding on the number of iterations (models) is critical. Too many numbers can reason for overfitting, while very few numbers can reason for underfitting. Validation methods help to choose the right number of iterations. Gradient boosting generally uses decision trees as base learners (Rokach, 2019).

XGBoost has some refinements and optimizations that are added to gradient boosting machines for increasing scalability. The key success factor of XGBoost is its scalability. The system operates more than ten times faster than available solutions on a machine and scales to the excessive number of examples in memory-limited or distributed settings. The property of scalability results from some innovations, including specific optimization techniques and systems. The proposed tree learning algorithm can handle sparse data with nodes' default directions; the algorithm uses a weighted quantile sketch procedure that provides handling weights of instances in approximate tree learning. Parallel and distributed computing contribute to the model's learning speed and exploration process. XGBoost benefits from the out-of-core computation. It uses a cache-aware structure that helps researchers process millions of instances on a desktop (Chen and Guestrin, 2016).

XGBoost includes split-finding algorithms that address weighted data by applying prune and merge operations, and all possible splits are enumerated efficiently, enabling optimize splitting threshold (Rokach, 2019).

A supervised learning algorithm generally refers to a mathematical equation in which prediction ($\hat{y}$) is made from inputs (x_i). The prediction of a linear model is given in equation (5.17). The weighted inputs are combined linearly. The prediction value is interpreted depending on the task, namely classification or regression. The aim of training a model is to find the parameters (Φ) that best fit the training data (x_i) and labels or target (y_i).

$$\hat{y}_i = \sum_j \Phi_j x_{ij} \tag{5.17}$$

The objective function of XGBoost consists of training loss and a regularization term. The objective function measures the fitness degree between the model and training data.

As one innovation, XGBoost attached regularization factor $\alpha(\Phi)$ to loss function for presenting tree complexity. The objective function of the algorithm is defined in equation 5.18, where Φ is a parameter trained from data. L represents the loss function of training, such as logistic loss. The loss function shows the goodness of fit between the given training data and the model. Depending on the problem, different loss functions can be used. $\alpha(\Phi)$ is a regularization term such as L1 regularization. This term controls (penalize) the complexity of the model and helps avoid overfitting.

$$\text{Obj}(\Phi) = \text{L}(\Phi) + \alpha(\Phi) \tag{5.18}$$

The prediction from a tree ensemble model is made by averaging T trees, using equation 5.19, in which the space of regression trees is shown as F.

$$\hat{y}_i = \sum_{t=1}^{T} f_t(x_i), f_t \epsilon F \tag{5.19}$$

The regularized objective function of the ensemble tree model is presented in equation 5.20, where L is a loss function, and α is the regularization term. m presents the number of predictions.

$$\text{Obj}(\Phi) = \sum_{i=1}^{m} \text{L}(y_i, \hat{y}_i) + \sum_{t=1}^{T} \alpha(f_t) \tag{5.20}$$

The regularization term is defined more specifically in equation 5.21, where β and θ are regularization factors. β controls the splitting of tree nodes. The node will split

when the loss function exceeds the β value.θ parameter is used to scale the penalty. M shows the number of leaves in a decision tree. w_n shows the leaf nodes' weight.

$$\alpha(f_t) = \beta M + \frac{1}{2}\theta \sum_{n=1}^{M} w_n^2 \tag{5.21}$$

XGBoost uses the second-order Taylor expansion in the optimization process. The objective function is expressed at step s according to equation 5.22, where g_i is first and h_i is the second derivative of the loss function.

$$Obj^s \approx \sum_{i=1}^{m} \left[L\left(y_i, \hat{y}_i^{(s-1)}\right) + g_i f_s(x_i) + \frac{1}{2} h_i f_s^2(x_i) \right] + \alpha(f_s) \tag{5.22}$$

$$= \sum_{i=1}^{m} \left[g_i f_i(x_i) + \frac{1}{2} h_i f_i^2(x_i) \right] + \alpha(f_s) + \sum_{i=1}^{m} L\left(y_i, \hat{y}_i^{(s-1)}\right)$$

The derivatives can be expressed as follow: $G_n = \sum_{i \epsilon I_n} g_i$ and $H_n = \sum_{i \epsilon I_n} h_i$, where I_n shows all the data instances in leaf node n. So objective function can be expressed in equation 5.23 (Li et al., 2020).

$$Obj^s = \sum_{n=1}^{M} \left[G_n w_n + \frac{1}{2}(H_n + \theta) w_n^2 \right] + \beta M \tag{5.23}$$

5.8.3 Bagging approach classifiers

Bagging Classifier, Random Forest, and Extra Trees are popular ensemble methods based on the bagging approach.

5.8.3.1 Bagging classifier

A bagging classifier is an ensemble-based method. Base classifiers fit each random subset of the original dataset. Then, classifiers' predictions (by applying a combination rule such as averaging, voting, etc.) are aggregated to make a final prediction. Such a type of meta estimator can be applied to decrease the variance of base estimators such as a decision tree by performing randomization into its building process and then making an ensemble out of it.

Bagging methods differ from each other depending on the way of drawing random samples from the training dataset. When replacement is done in drawing samples, the algorithm is called Bagging (Breiman, 1996). When random subsets are drawn as

random subsets of samples, the algorithm is called Pasting (Breiman, 1999). When random subsets are drawn as random subsets of features, the algorithm is called Random Subspaces (Ho, 1998). Lastly, the algorithm is called Random Patches, when base estimators are constructed on subsets of either features or samples (Louppe and Geurts, 2012).

The diversity in bagging is preserved by the variations among bootstrapped replicas where each classifier is trained, as well as by applying a notably weak classifier whose decision boundaries vary regarding relatively small disorders in training data. After the classifiers are trained, they are combined with a combination rule (Polikar, 2012).

The pseudo-code of the bagging classifier is presented in Table 5.6.

Table 5.6 : Pseudo code of typical bagging classifier.

Inputs: Training data (D), Base Classifier (BC), Number of classifiers (N), Percent (P) to generate bootstrapped training data
Do n= 1, 2, …., N Get a bootstrapped replica D_n by drawing randomly P% of D. Call BC with D_n and receive the classifier (hypothesis) C_n. Add C_n to ensemble, $\mathcal{E} \leftarrow \mathcal{E} \cup C_n$ End Combination Rule: such as majority voting, given unlabeled instance X Evaluate the ensemble $\mathcal{E} = \{C_1, C_2, \ldots, C_N\}$ on X. Let $v_{n,m}$ =1 if C_n chooses class w_m; otherwise, it is 0. Calculate total vote received by each class: $V_m = \sum_{n=1}^{N} v_{n,m}$, m=1,2,….M. Output: Class with the highest V_m.

This study uses decision trees as base estimators for the bagging classifier and is known as bagged decision trees. Moreover, Random Forest and Extra Trees algorithms use randomized decision trees.

5.8.3.2 Random forest

In bagged trees or any low-bias high variance technique, the predictive performance is improved by decreasing the variance of prediction over a single tree. Using bootstrapping resampling brings random factor in tree building, which leads to a distribution of predictions for each instance. The trees of a bagging model are not entirely independent of each other. The trees from varied bootstrap samples can present similar structures where similarity exists, particularly in the beginning of the trees. That means there is a correlation between trees, and this correlation inhibits the

bagging model from optimally decreasing the variance of predictions. So, there is a chance to improve the performance of a bagging model in terms of variance. Decreasing the correlation between trees (de-correlating trees) is a feasible way to enhance the performance of a bagging model. From the point of statistical terms, appending randomness to the tree's building process helps decrease in correlation among predictors (Kuhn and Johnson, 2013).

Several researchers fine-tuned the bagging algorithm by appending randomness in learning. Breiman (2001) developed the random forest algorithm, which attempts to decrease correlation between base learners by learning trees using a randomly selected subset of inputs and data cases (Murphy, 2012). It is an extension of the bagging of decision trees.

Random forest uses bagging on samples, majority voting, and random subsets. It can overcome missing data, various variables (such as categorical and continuous), and high dimensional data. Unlike usual decision trees, trees do not need to be pruned in random forest since they can overcome overfitting (Qi, 2012).

This algorithm offers extra randomness during the building trees. The algorithm looks for the best feature within a random subset of features instead of looking for the best features while splitting a node. The random forest algorithm leads to increased tree diversity that helps to obtain a lower variance and higher bias (Géron, 2019).

A basic random forest algorithm is presented in Table 5.7. Each decision tree is developed from a bootstrap sample of data. According to the algorithm, each model is used to make a prediction for a new sample, and obtained predictions (M number) are averaged to find out the prediction of the forest. The tree correlation will decrease because the random forest algorithm randomly chooses predictors (number of S) at each split. In the classification problems, each tree in the forest votes for a new sample classification similar to bagging (Kuhn and Johnson, 2013).

Random forest is better than bagging in computational efficiency. At each split, the random forest algorithm just evaluates a fraction from the original predictors, even though random forests generally need more trees. Additionally, combining this property with the parallel process in tree construction makes a random forest algorithm better than boosting in computational efficiency (Kuhn and Johnson, 2013).

Table 5.7 : Pseudo code of typical random forest classifier.

```
Decide on the number of models to construct, M
for k = 1 to M do
    Produce a bootstrap sample of training dataset
    Train a tree model on this sample
     for each split, do
          Randomly choose S (S < number of predictors) of original predictors
          Choose the best predictor between S predictors and partition/segment
the data (not prune)
     end
end
```

5.8.3.3 Extra trees

The Extra Trees is relevant to bagging and Random Forest Algorithm. It uses decision trees in the ensemble. In a Random Forest, during the building of a tree, at each node, just a random subset of features is taken into account for splitting. There is a possibility to build trees further randomly by applying random threshold values for each feature instead of looking for the best possible threshold values (like common decision trees make). In other words, nodes are split, selecting cut points randomly (Geurts et al., 2006; Géron, 2019). Such a forest is called Extremely Randomized Trees or Extra Trees, shortly. This algorithm offers a lower variance against more bias like the Random Forest. Extra Trees are faster than Random Forest because of the threshold strategy in building a tree. It is time-consuming to find the best likely threshold for each feature in every node during a tree building (Géron, 2019).

The Extra Trees generates lots of regression or decision trees that are unpruned from training data. The final prediction is made through aggregating predictions from trees. The majority voting is applied in classification, and averaging is applied in regression problems. Extra Trees also randomly samples features in every decision tree's split/separation point, similar to Random Forest. In addition to applying a random threshold, another difference from the other ensemble of tree algorithms is that Extra Trees uses the entire learning sample instead of using the bootstrap sample in growing the trees. Each decision tree is fitted to the entire training data. There are three important parameters in the tuning algorithm: number of trees, which affect the variance of model, number of attributes/input features chosen randomly at each node, which adjusts the robustness of the feature selection process, and minimum sample size to split a node, which adjusts robustness of averaging output noise (Geurts et al., 2006).

5.8.4 Logistic regression

Logistic regression (LR) is a binary classification model in machine learning. It is a supervised classifier. Like a linear regression model, the LR model includes an intercept (bias) and slope parameters for model terms. It takes real-valued input features and multiplies each of them a weight. Then summation is sent to the sigmoid function that gives a probability. This probability is compared with a threshold value to make a classification. LR model can also be used in multiple class cases, where the model is called multinomial LR. The softmax function is applied to obtain probabilities in a multinomial LR.

LR models an event's logarithm odds (log odds) using a linear function. The odds of an event are calculated by p/(1-p), where p is called the probability of an event. Log-odds is presented in equation 5.24, where n is the number of predictors.

$$\log\left(\frac{p}{1-p}\right) = b_0 + b_1x_1 + b_2x_2 + \cdots + b_nx_n \tag{5.24}$$

p (event probability function) can be obtained from Equations 5.24, as given in equation (5.25). This function is a sigmoidal function that is nonlinear.

$$p = \frac{1}{1 + \exp[-(b_0 + b_1x_1 + b_2x_2 + \cdots + b_nx_n)]} \tag{5.25}$$

In LR, a linear model is contained in a sigmoid, also called a logistic function. The logistic function is presented in Figure 5.14. The sigmoid function gets a real value and turns it into a value in the range of [0, 1].

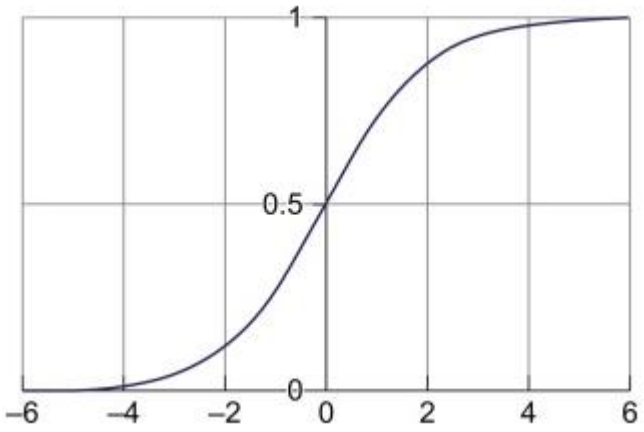


Figure 5.14 : Curve of logistic function.

A logistic function is presented in equation 5.26, where $P(y_m=1|X)$ gives the probability of the target value of m^{th} observation, being class 1. X represents training data; b_0 is the bias; (b_1,…, b_n) are weights that are adjusted for each predictor; Eular's number is shown as e. The logistic function gives an output between 0 and 1. When $P(y_m=1|X)$ is less than 0.5, class 0 is predicted; if not, class 1 is predicted (Albon, 2018).

$$P(y_m = 1 \mid X) = \frac{1}{1 + e^{-(b_0 + b_1 x_1 + b_2 x_2 + \cdots + b_n x_n)}} \tag{5.26}$$

The maximum likelihood estimation is the most common way of estimating the LR model parameters (b_0, b_1,…, b_n). The likelihood function (L) is given in equation 5.27, where m is the number of observations (Yang et al., 2020).

$$L = \prod_{i=1}^{m} P(Y = y_i \mid X = x_i) = \prod_{i=1}^{m} P(x_i)^{y_i}[1 - P(x_i)]^{1-y_i} \tag{5.27}$$

The estimated probability of the LR model can be presented in vectorized form, where σ is a sigmoid function (5.28). The model predicts 1 when $x^T\theta$ gets a positive value and predicts 0 when $x^T\theta$ gets a negative value.

$$\hat{p} = h_\theta(x) = \sigma(x^T\theta) \tag{5.28}$$

The aim of training the LR model is to adjust the θ (parameter vector), so the model estimates a high probability value for a positive instance and a low probability for a negative instance. The cost function is given for only an instance in equation 5.29, where C (θ) is the cost function.

$$C_i(\theta) = \begin{cases} -\log(1 - \hat{p}) & when\ y = 0 \\ -\log(\hat{p}) & when\ y = 1 \end{cases} \tag{5.29}$$

The cost function (average cost) for the whole training data is expressed in equation (5.30). It is called log loss.

$$C(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[\left(1 - y^i\right)\log\left(1 - \hat{p}^i\right) + y^i \log\left(\hat{p}^i\right)\right] \tag{5.30}$$

Because the cost function is convex, any optimization algorithm, such as gradient descent, can be applied to find the global minimum value (Géron, 2019).

5.8.5 Support vector machine

Support vector machines (SVMs) are supervised learning methods. They are impressive in high dimensional spaces. SVM is a generalization of an ordinary classifier named the maximal margin classifier. The maximal margin classifier is used for classes that can be separated linearly.

The support vector classifier (SVC) is an extension of the maximal margin classifier. It is also applied in linearly separable classes. SVC is called a soft margin classifier. It seeks a classifier using a hyperplane that considers more robustness to singular observations and preferable classification of most of the training data. It could be beneficial to make misclassification for a few training instances to make better classifications for the remaining instances. The SVM is a further extension of SVC. SVM enables to enlargement of feature space used by SVC by applying kernels. A kernel is a function that quantifies or evaluates the similarity of two observations. Using kernel functions, such as a polynomial kernel instead of a linear kernel, enables an SVC to build flexible decision boundaries (Gareth et al., 2013). A kernel function converts a nonlinear classification problem into a linear one in a high-dimensional space.

A training dataset is given in the form of $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$, where n is the number of points, y_i is either -1 or 1 that presents the class for each x_i point belongs. Each x_i is an N-dimensional real vector.

SVM is applied in a classification task according to Vapnik (1995) using the following equations (5.31):

$$\begin{cases} \boldsymbol{w}^T\theta(x_k) + \mathrm{b} \geq 1 \text{ if } y_k = 1 \\ \boldsymbol{w}^T\theta(x_k) + b \leq -1 \text{ if } y_k = -1 \end{cases} \tag{5.31}$$

Equation 5.31 is equivalent to equation (5.32).

$$y_k(\boldsymbol{w}^T\theta(\boldsymbol{x}_k) + \mathrm{b}) \geq 1, k = 1, 2, 3, \ldots, \mathrm{N} \tag{5.32}$$

Θ is a nonlinear function that maps the input space into a feature space that is high dimensional. A hyperplane ($w^T\theta(x_k) + \mathrm{b} = 0$) is formed to discriminate two classes. It is presented in Figure 5.15. Two parallel hyperplanes (dotted lines) separate two classes. The region between these hyperplanes is called as margin. During the model training, the minimum distance between training instances and hyperplanes, or

maximum margin, is searched to find optimal separation on linearly separable data (Wu et al., 2008).

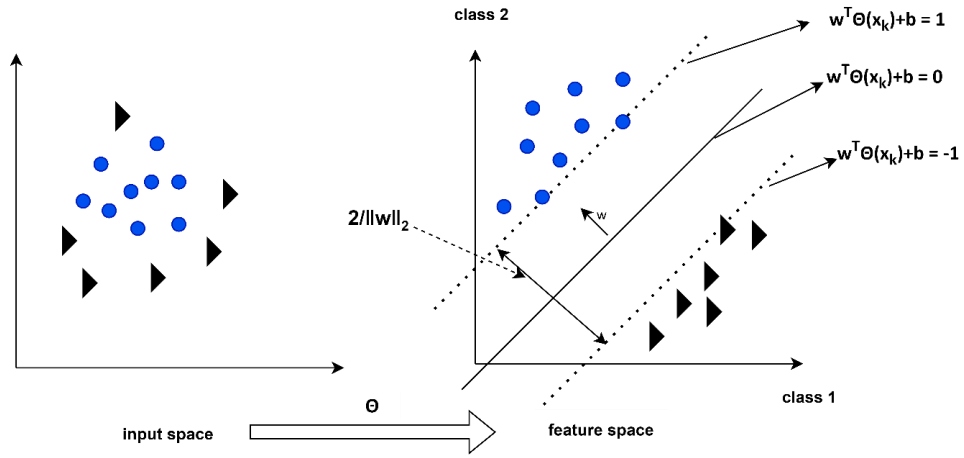


Figure 5.15 : Mapping input space (training data) into a higher dimensional space.

The decision function is presented in equation 5.33, where sign(.) is the sign function, and w is a normal vector to the hyperplane.

$$y(x) = \text{sign}(\boldsymbol{w}^T\theta(\mathbf{x}) + b) \tag{5.33}$$

The margin is calculated by $2/\|w\|_2$ where $\|w\|_2 = \sum_{k=1}^{n} w_k^2$. The $\|w\|_2$ is minimized subject to constraints in equation 5.32 for maximizing the margin. The classifier is called SVM because samples located on supporting planes (dotted lines) are called support vectors.

The optimization of an SVM classification problem can be presented in the following equations (5.34, 5.35):

$$\min_{w,b,\varepsilon} \quad F(\mathbf{w}, b, \varepsilon) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_{k=1}^{N}\varepsilon_k \tag{5.34}$$

$$\text{subject to} \quad \begin{cases} y_k(\boldsymbol{w}^T\theta(\boldsymbol{x}_k) + b) \geq 1 - \varepsilon_k, \\ \varepsilon_k \geq 0 \end{cases} \tag{5.35}$$

The variables ε_k are slack variables used to let misclassifications in inequalities. The left side in equation 5.34 undertakes to maximize the margin in feature space, while the right side tries to minimize the error of misclassification. C (constant) is a positive real value. It is used for tuning the algorithm.

Finally, the optimal discriminant function given in equation 5.36 is obtained by maximizing the dual function of equation 5.36 and Lagrangian multipliers, where k(.)

is a kernel function, and it is calculated by the inner product of vectors x and x_k in feature space; namely, $k(x, x_k) = \Theta(x)^T \Theta(x_k)$. Kernel function enables to make computations in low dimensional input space. α_k is the Lagrange multiplier, and x is the support vector (Wu et al., 2008).

$$y(x) = sign\left(\sum_{k=1}^{N} \alpha_k y_k k(\boldsymbol{x}, \boldsymbol{x}_k) + b\right) \tag{5.36}$$

5.8.6 Multilayer perceptron

Multilayer Perceptron (MLP) is a supervised learning algorithm and feedforward NN for function approximation.

An MLP consists of sequential layers: input layer, hidden or non-linear layer that can be one or more, and output layer. An f(.): $R^m \rightarrow R^n$ function is learned on a training dataset where m and n are the number of dimensions for input and output, respectively. Assume that $X = (x_1, x_2, \ldots, x_k)$ is a set of features and y is a target. MLP learns a function approximation that is non-linear for either regression or classification. Each neuron of the hidden layer transforms the values of the antecedent layer with a summation (weighted linear), and a nonlinear activation function follows it. Several functions can be applied as activation functions. The sigmoid or hyperbolic tangent functions are the most preferred ones (URL-15).

The output layer gives outputs related to the problem type by transforming the values from the last hidden layer. There are advantages and disadvantages of MLP. It has the capability of learning nonlinear models. An MLP model is sensitive to feature scaling. The loss function used is non-convex, where there are many local minimum points. So initializing the weights randomly can lead to different validation performances. The MLP is mostly trained with gradient descent, and the backpropagation algorithm is applied to calculate the gradients. The connection weights are adjusted iteratively to minimize an error function. In the classification tasks, the Cross-Entropy loss function is minimized. In regression tasks, the identity function is used in the output layer, and the square error is used as the loss function (URL-15). More details on training a NN are given in the ANN section.

A sample MLP network with one hidden layer is presented in Figure 5.16, where (x_1, x_2, …, x_k) is a set of inputs. Inputs are fed through neurons and multiplied by weights.

Then, bias is added to the sum of multiplications in each neuron. The summation is sent to the activation function (f_1), and a similar process is repeated to obtain the output (Sunthornnapha, 2017).

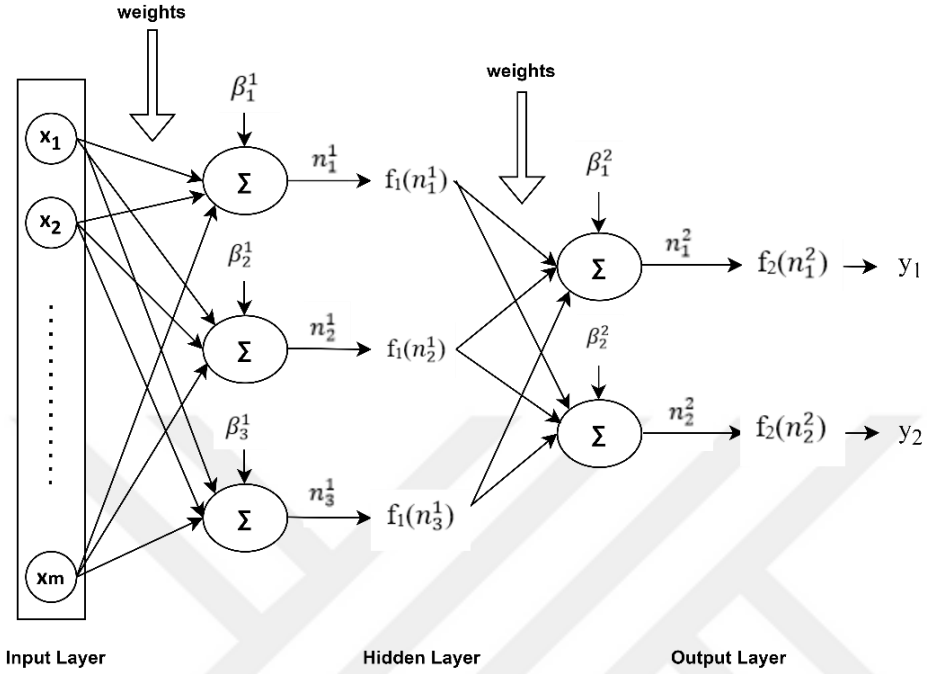


Figure 5.16 : A sample of one hidden layer MLP network.

The output y_i, (i=1, 2), of the MLP, is presented in equation (5.37).

$$y_i = f_2\left(\sum_{l=1}^{3} w_{li}^2 f_1\left(\sum_{m=1}^{M} w_{ml}^1 x_m + \beta_j^1\right) + \beta_j^2\right) \qquad (5.37)$$

6. EXPERIMENTAL ANALYSIS AND RESULTS

The analyses are made in Python and Matlab environments. The Tensorflow, Keras, Numpy, Scikit-learn, and XGBoost libraries are used. The computer has AMD Ryzen 7 3700X 8 core 36 MB processor and 8 GB graphics card, and 32 GB 3200 MHz DDR4 ram hardware sources.

This study randomly split images in the Messidor-2 dataset into training (80%) and testing data (20%).

In the first stage of the proposed model, images are pre-processed by resizing and pixel normalization. Pre-processing operations are considerably low.

The pre-processed images are fed into the InceptionV3. The layer connections of InceptionV3 from the input layer to the feature extraction layer are given in Table 6.1.

There are ten mixed (concatenate) layers throughout the InceptionV3. Features are extracted from the mixed-3 layer, one of the initial layers of the InceptionV3. The reason for choosing this layer is that the features extracted from mixed-3 give maximum accuracy with classifiers. Pre-trained, frozen weights of InceptionV3 on ImageNet are used.

In deep CNNs, spatial hierarchies of features are learned. Deeper layers encode high-level features (abstract features) such as a human nose, while initial layers work like edge detectors and detect simple shapes, lines, edges, etc., from the data. Since one of the initial layers is used, features obtained from the mixed-3 layer give more valuable information than a deeper layer.

Table 6.1 : Layer connections of InceptionV3 from input to mixed-3.

Layer (type)	Connected to
input layer	-
conv2d_1	input layer
batch_normalization_1	conv2d_1
activation_1	batch_normalization_1
conv2d_2	activation_1
batch_normalization_2	conv2d_2
activation_2	batch_normalization_2

Table 6.1 (continued): Layer connections of InceptionV3 from input to mixed-3.

Layer (type)	Connected to
conv2d_3	activation_2
batch_normalization_3	conv2d_3
activation_3	batch_normalization_3
max_pooling2d_1	activation_3
conv2d_4	max_pooling2d_1
batch_normalization_4	conv2d_4
activation_4	batch_normalization_4
conv2d_5	activation_4
batch_normalization_5	conv2d_5
activation_5	batch_normalization_5
max_pooling2d_2	activation_5
conv2d_9	max_pooling2d_2
batch_normalization_9	conv2d_9
activation_9	batch_normalization_9
conv2d_7	max_pooling2d_2
conv2d_10	activation_9
batch_normalization_7	conv2d_7
batch_normalization_10	conv2d_10
activation_7	batch_normalization_7
activation_10	batch_normalization_10
average_pooling2d_1	max_pooling2d_2
conv2d_6	max_pooling2d_2
conv2d_8	activation_7
conv2d_11	activation_10
conv2d_12	average_pooling2d_1
batch_normalization_6	conv2d_6
batch_normalization_8	conv2d_8
batch_normalization_11	conv2d_11
batch_normalization_12	conv2d_12
activation_6	batch_normalization_6
activation_8	batch_normalization_8
activation_11	batch_normalization_11
activation_12	batch_normalization_12
mixed0 (Concatenate)	activation_6 activation_8 activation_11 activation_12
conv2d_16	mixed0
batch_normalization_16	conv2d_16
activation_16	batch_normalization_16
conv2d_14	mixed0
conv2d_17	activation_16
batch_normalization_14	conv2d_14

Table 6.1 (continued): Layer connections of InceptionV3 from input to mixed-3.

Layer (type)	Connected to
batch_normalization_17	conv2d_17
activation_14	batch_normalization_14
activation_17	batch_normalization_17
average_pooling2d_2	mixed0
conv2d_13	mixed0
conv2d_15	activation_14
conv2d_18	activation_17
conv2d_19	average_pooling2d_2
batch_normalization_13	conv2d_13
batch_normalization_15	conv2d_15
batch_normalization_18	conv2d_18
batch_normalization_19	conv2d_19
activation_13	batch_normalization_13
activation_15	batch_normalization_15
activation_18	batch_normalization_18
activation_19	batch_normalization_19
mixed1 (Concatenate)	activation_13 activation_15 activation_18 activation_19
conv2d_23	mixed1
batch_normalization_23	conv2d_23
activation_23	batch_normalization_23
conv2d_21	mixed1
conv2d_24	activation_23
batch_normalization_21	conv2d_21
batch_normalization_24	conv2d_24
activation_21	batch_normalization_21
activation_24	batch_normalization_24
average_pooling2d_3	mixed1
conv2d_20	mixed1
conv2d_22	activation_21
conv2d_25	activation_24
conv2d_26	average_pooling2d_3
batch_normalization_20	conv2d_20
batch_normalization_22	conv2d_22
batch_normalization_25	conv2d_25
batch_normalization_26	conv2d_26
activation_20	batch_normalization_20
activation_22	batch_normalization_22
activation_25	batch_normalization_26
activation_26	batch_normalization_26

Table 6.1 (continued): Layer connections of InceptionV3 from input to mixed-3.

Layer (type)	Connected to
mixed2 (Concatenate)	activation_20 activation_22 activation_25 activation_26
conv2d_28	mixed2
batch_normalization_28	conv2d_28
activation_28	batch_normalization_28
conv2d_29	activation_28
batch_normalization_29	conv2d_29
activation_29	batch_normalization_29
conv2d_27	mixed2
conv2d_30	activation_29
batch_normalization_27	conv2d_27
batch_normalization_30	conv2d_30
activation_27	batch_normalization_27
activation_30	batch_normalization_30
max_pooling2d_3	mixed2
mixed3 (Concatenate)	activation_27 activation_30 max_pooling2d_3

After feature extraction, 3,251,712 (73×58×768) features are obtained from each fundus image. That means 768 feature maps in size of 73×58 are obtained from each image. These features are summarized by Global Average Pooling (GAP). The GAP is a downsampling operation that turns samples of an entire feature map into a single value by averaging the values. It summarizes the features in an image aggressively. Applying the GAP on extracted features gives 768 (1×1×768) features for each image.

GAP operation is illustrated in Figure 6.1. The 73×58 values or features in each feature map are averaged, and then obtained 768 features for each image are flattened to be used in the following steps: classification and feature selection.

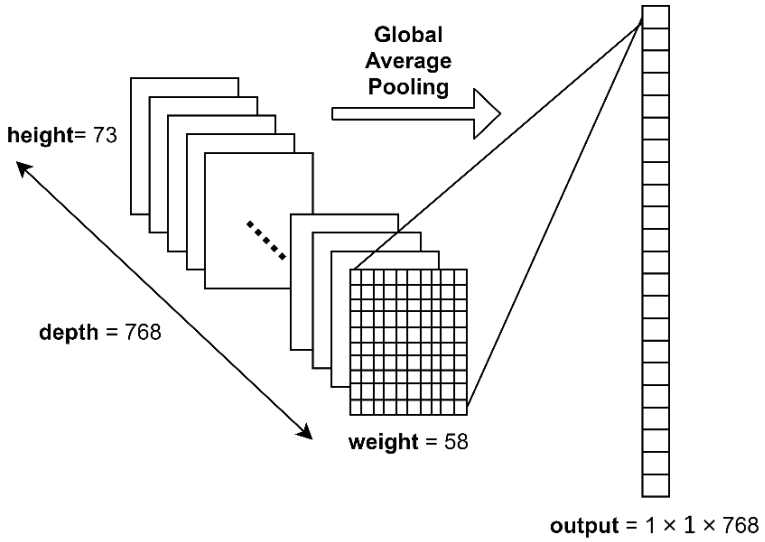


Figure 6.1 : GAP application.

Table 6.2 presents the comparison of classifiers in terms of accuracy level. Seven classifiers: XGBoost, Bagged Decision Trees, Extra Trees, Random Forest, LR, SVMs, and MLP are used. Binary classification is made: RDR versus NRDR. Grid search is carried out for classifiers. 5-fold cross-validation is applied to test the consistency of the model, and the results are consistent with the accuracy score obtained from 20% testing data so that the proposed model is reliable.

The Scikit-learn (URL-16) and XGBoost (URL-17) libraries are used, and parameters of algorithms except default ones are given in Table 6.2. XGBoost gave a maximum accuracy of 91.40%, followed by bagging approach classifiers. According to the results, ensemble methods are better than Logistic Regression, SVMs, and MLP.

Table 6.2 : Comparison of classifiers in binary classification with extracted features.

Classifier	Parameters	Accuracy (%)
XGBoost	lr = 0,1; noe = 400, max depth=4	91.40
Random Forest	noe = 100	89.68
Extra Trees	noe = 300	89.68
Bagged Decision Trees	noe = 20, be= decision tree	89.40
Support Vector Machines	probability=true, kernel=poly, degree=8	87.97
Logistic Regression	max_iter= 10000	87.39
Multilayer Perceptron	hidden_layer_sizes = (512,512); lr = adaptive	86.53

lr= learning rate; noe= number of estimators; be= base estimator.

The following process is feature selection with MHAs. The effective features are selected using PSO, SimAn, and ABC algorithms.

kNN classifier error rate is used as the fitness function for algorithms. For the kNN, euclidean distance is used as a distance metric, and the k-value (the number of neighbors) is determined by grid search and searched in the range of [2, 7]. A Uniform weight function is used where all points in each neighborhood are weighted equally. 5-fold cross-validation is made for the kNN.

Grid search is carried out for MHSs. Parameters of PSO are number of particles=20, maximum number of iterations=200, cognitive factor=2, social factor=2, maximum velocity=6, minimum bound on inertia weight=0.4, maximum bound on inertia weight=0.9.

Parameters of SimAn are annealing initial temperature=200, the cooling coefficient=0.999, and end temperature=1.

Parameters of ABC are: the population size is 20, the maximum number of iterations is 200, the abandonment limit is 100, the parameter's lower bound=-10, parameter's upper bound=10.

XGBoost s applied for the classification of selected features because of its superior performance over the other classifiers in experiments, as shown in Table 6.2.

Table 6.3 presents the XGBoost accuracy scores regarding the k value of kNN algorithm and the number of selected features for each MHA. The number of features is 768 before feature selection.

PSO algorithm gives maximum accuracy of 93.12% with selected 382 features when the k value is 7. SimAn provides maximum accuracy of 92.55% with selected 367 features when the k value is 5. ABC gives maximum accuracy of 91.69% with selected 360 features and 388 features when the k value is 5 and 7, respectively.

The MHAs increased the classification accuracy; simultaneously, the number of features decreased significantly.

Table 6.3 : XGBoost's accuracy for binary classification with selected features.

Fitness Function	Feature Selection Algorithms					
	ABC		SimAn		PSO	
k	# of feature	Accuracy %	# of feature	Accuracy %	# of feature	Accuracy %
2	382	90.54	367	90.83	357	90.83
3	399	91.12	411	90.83	378	91.98
4	366	91.12	381	91.69	387	89.40
5	360	91.69	367	92.55	363	91.40
6	367	90.83	381	89.68	389	90.26
7	388	91.69	401	90.83	382	93.12

k: the number of neighbors in kNN

Table 6.4 summarizes the overall classification results.

Table 6.4 : Classification of extracted and selected features with XGBoost.

Methods	Number of features	Accuracy (%) of XGBoost
Extracted Features from InceptionV3	768	91.40
Selected Features with PSO	382	93.12
Selected Features with SimAn	367	92.55
Selected Features with ABC	360	91.69

The confusion matrix of the maximum accuracy score, 93.12%, obtained from the combination of PSO feature selection and XGBoost classifier, is presented in Figure 6.2.

For the NRDR case, 274 out of 280 observations are classified correctly, while 51 out of 69 observations are classified correctly for the RDR case. The proposed model performs better for the NRDR case than RDR regarding misclassification error. The reason can be the difference between the number of training images for each case, which affects model performance.

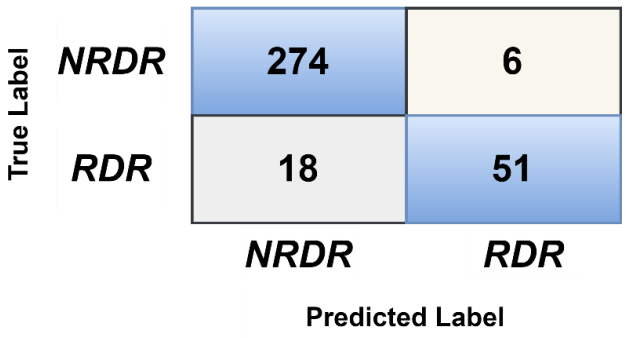


Figure 6.2 : Confusion matrix.

Some performance metrics are calculated from the confusion matrix. These are accuracy (Acc), specificity (SP), sensitivity (SE), F-score or F1 score, negative predictive value (NPV), and precision or positive predictive value (PR). The true negative (tn), true positive (tp), false negative (fn), and false positive (fp) values are used to calculate metrics. Formulas are given in equations 6.1-6.6. Metrics are calculated as follows: Acc of 93.12%, SP of 73.91%, SE of 97.85, F-score of 97.50%, NPV of 89.47%, and PR of 93.83%.

$$\mathrm{Acc} = \frac{\mathrm{tp} + \mathrm{tn}}{\mathrm{tp} + \mathrm{fp} + \mathrm{tn} + \mathrm{fn}} \tag{6.1}$$

$$\mathit{SP} = \frac{\mathrm{tn}}{\mathrm{tn} + \mathrm{fp}} \tag{6.2}$$

$$\mathrm{SE} = \frac{\mathrm{tp}}{\mathrm{tp} + \mathrm{fn}} \tag{6.3}$$

$$\mathrm{F-score} = \frac{2\mathrm{tp}}{2\mathrm{tp} + \mathrm{fp} + \mathrm{fn}} \tag{6.4}$$

$$\mathrm{NPV} = \frac{\mathrm{tn}}{\mathrm{tn} + \mathrm{fn}} \tag{6.5}$$

$$\mathrm{PR} = \frac{\mathrm{tp}}{\mathrm{tp} + \mathrm{fp}} \tag{6.6}$$

Table 6.5 gives a performance comparison of studies that uses Messidor-2 in validation.

The proposed model splits Messidor-2 into train and test sets, while the studies used in the comparison in Table 6.5 used different and considerably more image data in training. Additionally, because fine-tuning is not made, which increases training times considerably, the current study is not computationally expensive against the other studies that made fine-tuning. So the recent research offers competitive results.

Table 6.5 : Performance comparison of studies on Messidor-2.

Study	Method	Training Data (number of images)	Classes	Performance Measure
Gargeya and Leng (2017)	CNN + Gradient Boosting	EyePACS (75137)	No DR vs. any grade of DR	AUC of 94%
Li et al. (2019)	InceptionV3	Private (7935)	RDR vs. NRDR	Acc of 93.49%
Voets et al. (2019)	Ensemble of InceptionV3	EyePACS (45717)	RDR vs. NRDR	AUC of 85.3%
de La Torre et al. (2020)	Customized CNN	EyePACS (75650)	the most severe cases of DR vs. the rest	Acc of 91%
Toledo-Cortés et al. (2020)	InceptionV3 + Gaussian Process Regressor	EyePACS (56827)	RDR vs. NRDR	AUC of 87.87%
Saxena et al. (2020)	Inception and ResNet-based architectures	EyePACS (56839)	No DR vs. any grade of DR	AUC of 92%
Zago et al. (2020)	Patch-based CNN	DIARETDB1 (28)	RDR vs. NRDR	AUC of 94.4%
Tseng et al. (2020)	Fusion CNN architectures	Private (22617)	RDR vs. NRDR	Acc of 91.99%
The proposed study	InceptionV3 + Metaheuristic Algorithm +XGBoost	Messidor-2 (1392)	RDR vs. NRDR	Acc of 93.12%, AUC of 95.32%

7. CONCLUSIONS

Diabetic retinopathy is one of the leading causes of vision problems. Early detection and timely treatment prevent potential damage or vision loss in diabetes patients. Recently, computational methods: image processing techniques, machine learning, and deep learning-based models have been applied widely in DR diagnosis tasks. Our study's primary aim is to help the diagnosis process of DR in low-source settings.

This study gives a comprehensive literature review for DR classification where deep learning models are applied. A hybrid model is proposed. The Messidor-2 dataset is used in experiments. The model extracts retinal image representations from the InceptionV3, taking advantage of the transfer learning approach. The extracted features are classified with several machine learning algorithms, and the results are compared. Binary classification is made: NRDR vs. RDR.

Three metaheuristic algorithms, Particle Swarm Optimization, Simulated Annealing, and Artificial Bee Colony, are applied to select the best potential features from the extracted features. The selected features are classified with XGBoost. The feature selection with MHAs decreased the number of features besides increased accuracy performance.

This study proposes a novel hybrid architecture by taking advantage of a deep learning model and metaheuristic algorithms for DR classification. The number of parameters to be trained is considerably low, so the training time is short, and the model has few pre-processing steps. The model gives competitive results even with low computing power and a small number of images. These advantages make the model favorable for real-time applications. Since the model was proven reliable by considering different criteria, it can be used to provide decision support to ophthalmologists in diagnosing DR.

In this study, since the number of data is small, transfer learning approach is used, where frozen weights of a pre-trained model are used. In future studies, some layers of pre-trained deep learning models can be trained, or a customized CNN model can be proposed by using datasets containing more images. End-to-end deep learning models can be built since our study is a hybrid model.

In future studies, different metaheuristic algorithms can be used. A multi-class classification can be performed. The proposed model can be moved to a web or mobile-integrated platform.

REFERENCES

Abdel-Fatah, L., & Sangaiah, A. K. (2018). Metaheuristic algorithms: A comprehensive review. *Computational intelligence for multimedia big data on the cloud with engineering applications*, 185-231.

Abràmoff, M. D., Folk, J. C., Han, D. P., Walker, J. D., Williams, D. F., Russell, S. R., ... & Niemeijer, M. (2013). Automated analysis of retinal images for detection of referable diabetic retinopathy. *JAMA ophthalmology*, *131*(3), 351-357**.**

Agrawal, P., Abutarboush, H. F., Ganesh, T., & Mohamed, A. W. (2021). Metaheuristic Algorithms on Feature Selection: A Survey of One Decade of Research (2009-2019). *IEEE Access*, *9*, 26766-26791.

Albawi, S., Mohammed, T. A., & Al-Zawi, S. (2017). Understanding of a convolutional neural network. *In 2017 International Conference on Engineering and Technology (ICET*), August 1-6, IEEE.

Albon, C. (2018). *Machine learning with python cookbook: Practical solutions from preprocessing to deep learning*. O'Reilly Media, Inc.

Ali, R., Hardie, R. C., Narayanan, B. N., & Kebede, T. M. (2022). IMNets: Deep Learning Using an Incremental Modular Network Synthesis Approach for Medical Imaging Applications. *Applied Sciences*, *12*(11), 5500.

Altman, N. S. (1992). An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, *46*(3), 175-185.

Anaya-Isaza, A., Mera-Jiménez, L., & Zequera-Diaz, M. (2021). An overview of deep learning in medical imaging. *Informatics in Medicine Unlocked*, *26*, 100723.

Arora, S., Bhaskara, A., Ge, R., & Ma, T. (2014). Provable bounds for learning some deep representations. *In International conference on machine learning*, January (pp. 584-592). PMLR.

Ayala, A., Ortiz Figueroa, T., Fernandes, B., & Cruz, F. (2021). Diabetic Retinopathy Improved Detection Using Deep Learning. *Applied Sciences*, *11*(24), 11970.

Aydoğan, E. K., Delice, Y., Özcan, U., Gencer, C., & Bali, Ö. (2019). Balancing stochastic U-lines using particle swarm optimization. *Journal of Intelligent Manufacturing*, *30*(1), 97-111.

Blum, C., & Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM computing surveys (CSUR)*, *35*(3), 268-308.

Bodapati, J. D., Naralasetti, V., Shareef, S. N., Hakak, S., Bilal, M., Maddikunta, P. K. R., & Jo, O. (2020). Blended multi-modal deep convnet features for diabetic retinopathy severity prediction. *Electronics*, *9*(6), 914.

Bodapati, J. D., Shaik, N. S., & Naralasetti, V. (2021). Composite deep neural network with gated-attention mechanism for diabetic retinopathy severity classification. *Journal of Ambient Intelligence and Humanized Computing, 12*(10), 9825-9839.

Bowling, B. (2016). Kanski's clinical ophthalmology. Edinburgh: Elsevier.

Breiman, L. (1996). Bagging predictors. *Machine learning, 24*(2), 123-140.

Breiman, L. (1999). Pasting small votes for classification in large databases and on-line. *Machine learning, 36*(1), 85-103.

Breiman, L. (2001). Random forests. *Machine learning, 45*(1), 5-32.

Brereton, R. G., & Lloyd, G. R. (2010). Support vector machines for classification and regression. *Analyst, 135*(2), 230-267.

Brownlee, J. (2011). Clever algorithms: nature-inspired programming recipes. Retrieved from https://machinelearningmastery.com/products/

Brownlee, J. (2016). XGBoost with Python. Retrieved from https://machinelearningmastery.com/products/

Canayaz, M. (2021). MH-COVIDNet: Diagnosis of COVID-19 using deep neural networks and meta-heuristic-based feature selection on X-ray images. *Biomedical Signal Processing and Control, 64*, 102257.

Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 785-794.

Chen, C., Liaw, A., & Breiman, L. (2004). *Using random forest to learn imbalanced data. University of California*, Berkeley, *110*(1-12), 24.

Chollet, F. (2017). *Deep learning with Python*. Simon and Schuster.

Çakmak, E., Önden, İ., Acar, A. Z., & Eldemir, F. (2021). Analyzing the location of city logistics centers in Istanbul by integrating Geographic Information Systems with Binary Particle Swarm Optimization algorithm. *Case Studies on Transport Policy, 9*(1), 59-67.

Darwish, A., Sayed, G., & Hassanien, A. (2018). Meta-Heuristic Optimization Algorithms Based Feature Selection for Clinical Breast Cancer Diagnosis. *Journal of the Egyptian Mathematical Society, 26*(2), 321-336.

Das, S., Kharbanda, K., Suchetha, M., Raman, R., & Dhas, E. (2021). Deep learning architecture based on segmented fundus image features for classification of diabetic retinopathy. *Biomedical Signal Processing and Control, 68*, 102600.

Decencière, E., Zhang, X., Cazuguel, G., Lay, B., Cochener, B., Trone, C., ... & Klein, J. C. (2014). Feedback on a publicly distributed image database: the Messidor database. *Image Analysis & Stereology, 33*(3), 231-234.

de La Torre, J., Valls, A., & Puig, D. (2020). A deep learning interpretable classifier for diabetic retinopathy disease grading. *Neurocomputing, 396*, 465-476.

Doshi, N., Oza, U., & Kumar, P. (2020). Diabetic retinopathy classification using downscaling algorithms and deep learning. In *2020 7th International Conference on Signal Processing and Integrated Networks (SPIN)*, IEEE, 950-955.

Engelbrecht, A. P. (2007). *Computational intelligence: an introduction*. John Wiley & Sons.

Ezugwu, A. E., Shukla, A. K., Nath, R., Akinyelu, A. A., Agushaka, J. O., Chiroma, H., & Muhuri, P. K. (2021). Metaheuristics: a comprehensive overview and classification along with bibliometric analysis. *Artificial Intelligence Review*, 1-80.

Farooq, M. S., Arooj, A., Alroobaea, R., Baqasah, A. M., Jabarulla, M. Y., Singh, D., & Sardar, R. (2022). Untangling computer-aided diagnostic system for screening diabetic retinopathy based on deep learning techniques. *Sensors*, *22*(5), 1803.

Fix, E., & Hodges Jr, J. L. (1951). *Discriminatory Analysis-Nonparametric Discrimination: Consistency Properties*. California Univ Berkeley.

Gareth, J., Daniela, W., Trevor, H., & Robert, T. (2013). *An introduction to statistical learning: with applications in R*. Spinger.

Gao, Z., Li, J., Guo, J., Chen, Y., Yi, Z., & Zhong, J. (2018). Diagnosis of diabetic retinopathy using deep neural networks. *IEEE Access*, *7*, 3360-3370.

Gargeya, R., & Leng, T. (2017). Automated identification of diabetic retinopathy using deep learning. *Ophthalmology*, *124*(7), 962-969.

Géron, A. (2019). *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. O'Reilly Media.

Geurts, P., Ernst, D., & Wehenkel, L. (2006). Extremely randomized trees. *Machine learning*, *63*(1), 3-42.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press.

Gulshan, V., Peng, L., Coram, M., Stumpe, M. C., Wu, D., Narayanaswamy, A., ... & Webster, D. R. (2016). Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs. *Jama*, *316*(22), 2402-2410.

Gunavathi, C., & Premalatha, K. (2015). Cuckoo search optimisation for feature selection in cancer classification: a new approach. *International journal of data mining and bioinformatics*, *13*(3), 248-265.

Hagos, M. T., & Kant, S. (2019). *Transfer learning based detection of diabetic retinopathy from small dataset*. arXiv preprint arXiv:1905.07203.

Hastie, T., Tibshirani, R., & Friedman, J. H. (2009). *The elements of statistical learning: data mining, inference, and prediction*. 2nd ed. New York: Springer.

Hazim, J. M., Hassan, H. A., Yassin, A. I. M., Tahir, N. M., Zabidi, A., Rizman, Z. I., ... & Wahab, N. A. (2018). Early detection of diabetic retinopathy by using deep learning neural network. *International Journal of Engineering & Technology*, *7*(4.11), 198-201.

Ho, T. K. (1998). The random subspace method for constructing decision forests. *IEEE transactions on pattern analysis and machine intelligence*, *20*(8), 832-844.

Hussain, K., Salleh, M. N. M., Cheng, S., & Shi, Y. (2019). Metaheuristic research: a comprehensive survey. *Artificial Intelligence Review*, *52*(4), 2191-2233.

Imran, M., Ullah, A., Arif, M., & Noor, R. (2022). A unified technique for entropy enhancement based diabetic retinopathy detection using hybrid neural network. *Computers in Biology and Medicine*, 105424.

Jiang, F., Xia, H., Tran, Q. A., Ha, Q. M., Tran, N. Q., & Hu, J. (2017). A new binary hybrid particle swarm optimization with wavelet mutation. *Knowledge-Based Systems*, *130*, 90-101.

Kandel, I., & Castelli, M. (2020). Transfer learning with convolutional neural networks for diabetic retinopathy image classification. A review. *Applied Sciences*, *10*(6), 2021.

Karaboga, D. (2005). *An idea based on honey bee swarm for numerical optimization* (Vol. 200, pp. 1-10). Technical report-tr06, Erciyes university, engineering faculty, computer engineering department.

Kassani, S. H., Kassani, P. H., Khazaeinezhad, R., Wesolowski, M. J., Schneider, K. A., & Deters, R. (2019, December). Diabetic retinopathy classification using a modified xception architecture. In *2019 IEEE international symposium on signal processing and information technology (ISSPIT)*, IEEE, pp. 1-6.

Kim, P. (2017). *Matlab deep learning with machine learning, neural networks and artificial intelligence*, Apress, New York.

Kim, D. W., Kim, K. H., Jang, W., & Chen, F. F. (2002). Unrelated parallel machine scheduling with setup times using simulated annealing. *Robotics and Computer-Integrated Manufacturing*, *18*(3-4), 223-231.

Kochenderfer, M. J., & Wheeler, T. A. (2019). *Algorithms for optimization.* Mit Press.

Kuhn, M., & Johnson, K. (2013). *Applied predictive modeling* (Vol. 26, p. 13). New York: Springer.

Kumari, K., Singh, J. P., Dwivedi, Y. K., & Rana, N. P. (2021). Multi-modal aggression identification using convolutional neural network and binary particle swarm optimization. *Future Generation Computer Systems*, *118*, 187-197.

Lakshminarayanan, V., Kheradfallah, H., Sarkar, A., & Jothi Balaji, J. (2021). Automated Detection and Diagnosis of Diabetic Retinopathy: A Comprehensive Survey. *Journal of Imaging*, *7*(9), 165.

Lam, C., Yi, D., Guo, M., & Lindsey, T. (2018). Automated detection of diabetic retinopathy using deep learning. *AMIA summits on translational science proceedings*, 147-155.

Lecun, Y. A., Bottou, L., Orr, G. B., & Müller, K. R. (2012). *Efficient backprop. In Neural networks: Tricks of the trade* (pp. 9-48). Springer, Berlin, Heidelberg.

Lee, H. T., Lee, J. S., Son, W. J., & Cho, I. S. (2020). Development of Machine Learning Strategy for Predicting the Risk Range of Ship's Berthing Velocity. *Journal of Marine Science and Engineering*, *8*(5), 376.

Li, F., Liu, Z., Chen, H., Jiang, M., Zhang, X., & Wu, Z. (2019). Automatic detection of diabetic retinopathy in retinal fundus photographs based on deep learning algorithm. *Translational vision science & technology*, *8*(6), 4-4.

Li, S., Qin, J., He, M., & Paoli, R. (2020). Fast evaluation of aircraft icing severity using machine learning based on XGBoost. *Aerospace*, *7*(4), 36.

Li, X., Pang, T., Xiong, B., Liu, W., Liang, P., & Wang, T. (2017). Convolutional neural networks based transfer learning for diabetic retinopathy fundus image classification. In *2017 10th international congress on image and signal processing, biomedical engineering and informatics (CISP-BMEI)*, IEEE, 1-11.

Lin, Y., Wang, J., Li, X., Zhang, Y., & Huang, S. (2021). An Improved Artificial Bee Colony for Feature Selection in QSAR. *Algorithms*, *14*(4), 120.

Liu, H., Motoda, H., Setiono, R., & Zhao, Z. (2010). Feature selection: An ever evolving frontier in data mining. In *Feature selection in data mining* (pp. 4-13). PMLR.

Liu, H., & Zhao, Z. (2012). Manipulating data and dimension reduction methods: Feature selection. In *Computational Complexity: theory, techniques, and applications* (pp. 1790-1800). Springer New York.

Louppe, G., & Geurts, P. (2012). Ensembles on random patches. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases* (pp. 346-361). Springer, Berlin, Heidelberg.

Mateen, M., Wen, J., Song, S., & Huang, Z. (2019). Fundus image classification using VGG-19 architecture with PCA and SVD. *Symmetry*, *11*(1), 1.

Meenachi, L., & Ramakrishnan, S. (2021). Metaheuristic Search Based Feature Selection Methods for Classification of Cancer. *Pattern Recognition*, *119*, 108079.

Mohamed, A. W., Hadi, A. A., & Mohamed, A. K. (2020). Gaining-sharing knowledge based algorithm for solving optimization problems: a novel nature-inspired algorithm. *International Journal of Machine Learning and Cybernetics*, *11*(7), 1501-1529.

Mohammadian, S., Karsaz, A., & Roshan, Y. M. (2017). Comparative study of fine-tuning of pre-trained convolutional neural networks for diabetic retinopathy screening. In *2017 24th National and 2nd International Iranian Conference on Biomedical Engineering (ICBME)* (pp. 1-6). IEEE.

Murphy, K. P. (2012). *Machine learning: a probabilistic perspective*. MIT press.

Nagpal, D., Panda, S. N., Malarvel, M., Pattanaik, P. A., & Khan, M. Z. (2021). A review of diabetic retinopathy: Datasets, approaches, evaluation metrics and future trends. *Journal of King Saud University-Computer and Information Sciences*.

Navot, A., Shpigelman, L., Tishby, N., & Vaadia, E. (2005). Nearest neighbor based feature selection for regression and its application to neural activity. *Advances in neural information processing systems*, *18*, 996-1002.

Nneji, G. U., Cai, J., Deng, J., Monday, H. N., Hossin, M. A., & Nahar, S. (2022). Identification of Diabetic Retinopathy Using Weighted Fusion Deep Learning Based on Dual-Channel Fundus Scans. *Diagnostics*, *12*(2), 540.

Nivrito, A. K. M., Wahed, M., & Bin, R. (2016). *Comparative analysis between InceptionV3 and other learning systems using facial expressions detection* (Doctoral dissertation, BRAC University).

Orlando, J. I., Prokofyeva, E., Del Fresno, M., & Blaschko, M. B. (2018). An ensemble deep learning based approach for red lesion detection in fundus images. *Computer methods and programs in biomedicine*, *153*, 115-127.

Paradisa, R. H., Bustamam, A., Mangunwardoyo, W., Victor, A. A., Yudantha, A. R., & Anki, P. (2021). Deep Feature Vectors Concatenation for Eye Disease Detection Using Fundus Image. Electronics, *11*(1), 23.

Peng, C. Y. J., Lee, K. L., & Ingersoll, G. M. (2002). An introduction to logistic regression analysis and reporting. *The journal of educational research*, *96*(1), 3-14.

Polikar, R. (2012). *Ensemble learning. In Ensemble machine learning (pp. 1-34).* Springer, Boston, MA.

Puspadini, R., Mawengkang, H., & Efendi, S. (2020). Feature Selection on K-Nearest Neighbor Algorithm Using Similarity Measure. *In 2020 3rd International Conference on Mechanical, Electronics, Computer, and Industrial Technology (MECnIT)* (pp. 226-231). IEEE.

Qi, Y. (2012*). Random forest for bioinformatics. In Ensemble machine learning* (pp. 307-323). Springer, Boston, MA.

Rajabi Moshtaghi, H., Toloie Eshlaghy, A., & Motadel, M. R. (2021). A comprehensive review on meta-heuristic algorithms and their classification with novel approach. *Journal of Applied Research on Industrial Engineering*, *8*(1), 63-89.

Resnikoff, S., Lansingh, V. C., Washburn, L., Felch, W., Gauthier, T. M., Taylor, H. R., ... & Wiedemann, P. (2020). Estimated number of ophthalmologists worldwide (International Council of Ophthalmology update): will we meet the needs?. *British Journal of Ophthalmology*, *104*(4), 588-592.

Ribeiro, M. H. D. M., & dos Santos Coelho, L. (2020). Ensemble approach based on bagging, boosting and stacking for short-term prediction in agribusiness time series. *Applied Soft Computing*, *86*, 105837.

Rokach, L. (2019). *Ensemble learning: Pattern classification using ensemble methods*. World Scientific.

Rosebrock, A. (2017). *Deep learning for computer vision with Python: starter bundle*. PyImageSearch.

Sahlsten, J., Jaskari, J., Kivinen, J., Turunen, L., Jaanio, E., Hietala, K., & Kaski, K. (2019). Deep learning fundus image analysis for diabetic retinopathy and macular edema grading. *Scientific reports, 9*(1), 1-11.

Sarı, M., Ahmad, A. A., & Uslu, H. (2021). Medical model estimation with particle swarm optimization. *Communications Faculty of Sciences University of Ankara Series A1 Mathematics and Statistics, 70*(1), 468-482.

Saxena, G., Verma, D. K., Paraye, A., Rajan, A., & Rawat, A. (2020). Improved and robust deep learning agent for preliminary detection of diabetic retinopathy using public datasets. *Intelligence-Based Medicine, 3*, 100022.

Soler-Dominguez, A., Juan, A. A., & Kizys, R. (2017). A survey on financial applications of metaheuristics. *ACM Computing Surveys (CSUR), 50*(1), 1-23.

Sörensen, K. (2015). Metaheuristics—the metaphor exposed. *International Transactions in Operational Research, 22*(1), 3-18.

Spall, J. C. (2012). *Stochastic optimization. In Handbook of computational statistics* (pp. 173-201). Springer, Berlin, Heidelberg.

Stegherr, H., Heider, M., & Hähner, J. (2020). Classifying Metaheuristics: Towards a unified multi-level classification system. *Natural Computing*, 1-17.

Stojanović, I., Brajević, I., Stanimirović, P. S., Kazakovtsev, L. A., & Zdravev, Z. (2017). Application of heuristic and metaheuristic algorithms in solving constrained weber problem with feasible region bounded by arcs. *Mathematical Problems in Engineering*.

Sunthornnapha, T. (2017). Utilization of MLP and linear regression methods to build a reliable energy baseline for self-benchmarking evaluation. Energy *Procedia, 141*, 189-193.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1-9).

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2818-2826).

Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. A. (2017). Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-first AAAI conference on artificial intelligence*.

Tahir, M. A., Bouridane, A., & Kurugollu, F. (2007). Simultaneous feature selection and feature weighting using Hybrid Tabu Search/K-nearest neighbor classifier. *Pattern Recognition Letters, 28*(4), 438-446.

Takahashi, H., Tampo, H., Arai, Y., Inoue, Y., & Kawashima, H. (2017). Applying artificial intelligence to disease staging: Deep learning for improved staging of diabetic retinopathy. *PloS one, 12*(6), e0179790.

Tan, C., Sun, F., Kong, T., Zhang, W., Yang, C., & Liu, C. (2018, October). A survey on deep transfer learning. In *International conference on artificial neural networks* (pp. 270-279). Springer, Cham.

Tariq, H., Rashid, M., Javed, A., Zafar, E., Alotaibi, S. S., & Zia, M. Y. I. (2021). Performance Analysis of Deep-Neural-Network-Based Automatic Diagnosis of Diabetic Retinopathy. *Sensors, 22*(1), 205.

Teo, Z. L., Tham, Y. C., Yu, M., Chee, M. L., Rim, T. H., Cheung, N., ... & Cheng, C. Y. (2021). Global prevalence of diabetic retinopathy and projection of burden through 2045: systematic review and meta-analysis. *Ophthalmology, 128*(11), 1580-1591.

Teuwen, J., & Moriakov, N. (2020). Convolutional neural networks. *In Handbook of medical image computing and computer assisted intervention* (pp. 481-501). Academic Press.

Toledo-Cortés, S., De La Pava, M., Perdómo, O., & González, F. A. (2020). Hybrid deep learning gaussian process for diabetic retinopathy diagnosis and uncertainty quantification. In *International Workshop on Ophthalmic Medical Image Analysis* (pp. 206-215). Springer, Cham.

Tseng, V. S., Chen, C. L., Liang, C. M., Tai, M. C., Liu, J. T., Wu, P. Y., ... & Chen, Y. H. (2020). Leveraging multi-modal deep learning architecture with retina lesion information to detect diabetic retinopathy. *Translational Vision Science & Technology, 9*(2), 41-41.

Tymchenko, B., Marchenko, P., & Spodarets, D. (2020*). Deep learning approach to diabetic retinopathy detection*. arXiv preprint arXiv:2003.02261.

URL-1<*https://machinelearningmastery.com/calculus-in-action-neural-networks/*>, date retrieved 07.06.2022.

URL-2<https://www.mathworks.com/discovery/deep-learning.html>, date retrieved 21.08.2021.

URL-3<*https://machinelearningmastery.com/convolutional-layers-for-deep-learning-neural-networks/*>, date retrieved 03.02.2022.

URL-4<*https://machinelearningmastery.com/how-to-use-transfer-learning-when-developing-convolutional-neural-network-models/*>, date retrieved 08.05.2022.

URL-5 <*https://www.adcis.net/en/third-party/messidor/*>, date retrieved 07.06.2021.

URL-6 <*https://keras.io/api/applications/*>, date retrieved 26.05.2021.

URL-7 <*https://www.image-net.org/*>, date retrieved 01.07.2021.

URL-8<*https://deepai.org/machine-learning-glossary-and-terms/inception-module*>, date retrieved 20.05.2021.

URL-9<*https://cloud.google.com/tpu/docs/InceptionV3-advanced*>, date retrieved 13.05.2021.

URL-10<*https://scikit-learn.org/stable/modules/neighbors.html#nearest-neighbors classification*>, date retrieved 06.06.2021.

URL11<*https://www.mathworks.com/help/stats/classificationknn.loss.html#btar64m-2_head*>, date retrieved 09.03.2022.

URL-12 < *https://abc.erciyes.edu.tr/*>, date retrieved 12.01.2022.

URL-13<*https://scikit-learn.org/stable/modules/ensemble.html#*>, date retrieved 23.04.2021.

URL-14<*https://machinelearningmastery.com/stacking-ensemble-machine-learning-with-python/*>, date retrieved 06.06.2021.

URL-15<*https://scikit-learn.org/stable/modules/neural_networks_supervised.html*>, date retrieved 23.04.2021.

URL-16<*https://scikit-learn.org/stable/index.html*>, date retrieved 15.09.2021.

URL-17<*https://xgboost.readthedocs.io/en/stable/index.html#*>, date retrieved 26.05.2021.

Uzer, M. S., Yilmaz, N., & Inan, O. (2013). Feature selection method based on artificial bee colony algorithm and support vector machines for medical datasets classification. *The Scientific World Journal*.

Vapink, V. N. (1995). *The nature of statistical learning theory*. M. Springer, Verlag.

Vo, H. H., & Verma, A. (2016). New deep neural nets for fine-grained diabetic retinopathy recognition on hybrid color space. In *2016 IEEE International Symposium on Multimedia (ISM)* (pp. 209-215). IEEE.

Voets, M., Møllersen, K., & Bongo, L. A. (2019). Reproduction study using public data of: Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs. *PloS one, 14*(6), e0217541.

Wan, S., Liang, Y., & Zhang, Y. (2018). Deep convolutional neural networks for diabetic retinopathy detection by image classification. *Computers & Electrical Engineering, 72*, 274-282.

Wang, L., Ni, H., Yang, R., Pappu, V., Fenn, M. B., & Pardalos, P. M. (2014). Feature selection based on meta-heuristics for biomedicine. *Optimization Methods and Software, 29*(4), 703-719.

Wen, L., & Hughes, M. (2020). Coastal wetland mapping using ensemble learning algorithms: A comparative study of bagging, boosting and stacking techniques. *Remote Sensing, 12*(10), 1683.

Witten, I. H., Frank, E., Hall, M. A., & Pal, C. J. (2016*). Data Mining: Practical Machine Learning Tools and Techniques*, Morgan Kaufmann.

Wu, T. K., Huang, S. C., & Meng, Y. R. (2008). Evaluation of ANN and SVM classifiers as predictors to the diagnosis of students with learning disabilities. *Expert Systems with Applications, 34*(3), 1846-1856.

Wu, C. C., Hsu, P. H., & Lai, K. (2011). Simulated-annealing heuristics for the single-machine scheduling problem with learning and unequal job release times. *Journal of Manufacturing Systems, 30*(1), 54-62.

Xue, B., Zhang, M., Browne, W. N., & Yao, X. (2015). A survey on evolutionary computation approaches to feature selection. *IEEE Transactions on Evolutionary Computation, 20*(4), 606-626.

Yamashita, R., Nishio, M., Do, R. K. G., & Togashi, K. (2018). Convolutional neural networks: an overview and application in radiology. *Insights into imaging, 9*(4), 611-629.

Yang, Y., Chen, G., & Reniers, G. (2020). Vulnerability assessment of atmospheric storage tanks to floods based on logistic regression. *Reliability Engineering & System Safety, 196*, 106721.

Yi, S. L., Yang, X. L., Wang, T. W., She, F. R., Xiong, X., & He, J. F. (2021). Diabetic Retinopathy Diagnosis Based on RA-EfficientNet. *Applied Sciences, 11*(22), 11035.

Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How transferable are features in deep neural networks?. *Advances in neural information processing systems, 27*.

Yusta, S. C. (2009). Different metaheuristic strategies to solve the feature selection problem. *Pattern Recognition Letters, 30*(5), 525-534.

Zago, G. T., Andreão, R. V., Dorizzi, B., & Salles, E. O. T. (2020). Diabetic retinopathy detection using red lesion localization and convolutional neural networks. *Computers in biology and medicine, 116*, 103537.

Zeng, X., Chen, H., Luo, Y., & Ye, W. (2019). Automated diabetic retinopathy detection based on binocular siamese-like convolutional neural network. *IEEE Access, 7*, 30744-30753.

Zhang, W., Zhong, J., Yang, S., Gao, Z., Hu, J., Chen, Y., & Yi, Z. (2019). Automated identification and grading system of diabetic retinopathy using deep neural networks. *Knowledge-Based Systems, 175*, 12-25.

Zhou, Z. H. (2019). *Ensemble methods: foundations and algorithms*. Chapman and Hall/CRC.

CURRICULUM VITAE

Name Surname **:** Ömer Faruk GÜRCAN

EDUCATION **:**

- **B.Sc.** **:** 2011, Selçuk University, Engineering Faculty, Industrial Engineering Department
- **M.Sc.** **:** 2014, İstanbul Technical University, Management Faculty, Industrial Engineering Department

PUBLICATIONS, PRESENTATIONS AND PATENTS ON THE THESIS:

- **Gürcan, Ö. F.,** Beyca, Ö. F., Doğan, O. 2021. A Comprehensive Study Of Machine Learning Methods On Diabetic Retinopathy Classification, *International Journal of Computational Intelligence Systems*, 14(1), 1132-1141.