جامعة أبو بكر بلقايد – تلمسان

Université Abou Bakr Belkaïd de Tlemcen

Faculté de Technologie

Département de Génie Biomédical

MEMOIRE DE PROJET DE FIN D'ETUDES

pour l'obtention du Diplôme de

MASTER en GENIE BIOMEDICAL

Spécialité : **Informatique Biomédicale**

présenté par : BRIKI Mohamed Elamine

Medical images classification based on deep features extraction exploiting transfer learning

Soutenu le 28 Novembre 2020 devant le Jury

M.	EL HABIB DAHO Mostafa	*MCB*	Université de Tlemcen	Président
M.	BEHADADA Omar	*MCB*	Université de Tlemcen	Examinateur
Mme	SETTOUTI Nesma	*MCA*	Université de Tlemcen	Encadreur
M.	BECHAR Mohammed El Amine	*MCB*	Université de Tlemcen	Co-encadreur

Année universitaire 2019-2020

MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR
ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITÉ ABOU BEKR BELKAID
FACULTÉ DE TECHNOLOGIE
DÉPARTEMENT DE GÉNIE BIOMÉDICAL

MÉMOIRE DE FIN D'ÉTUDES

pour obtenir le grade de

MASTER EN GÉNIE BIOMÉDICAL

Spécialité : **Informatique Biomédicale**

présenté et soutenu publiquement
par

Mr. Mohamed Elamine BRIKI

le 28 Novembre 2020

Titre:

Medical images classification based on deep features extraction exploiting transfer learning

Jury

Président du jury.	Dr. EL HABIB DAHO Mostafa,	MCB UABB Tlemcen
Examinateur.	Dr. BEHADADA Omar,	MCB UABB Tlemcen
Directrice de mémoire.	Dr. SETTOUTI Nesma,	MCA UABB Tlemcen
Co-Directeur de mémoire.	Dr. BECHAR Mohammed El Amine,	MCB UABB Tlemcen

A special dedication goes to :

***My parents**, who sustained me day and night in all ways and literary made me what I am today,*

***My family** who prayed for my success and supported me all along,*

***my professors** who sincerely dedicated their time and effort to teach us class/life lessons and inspired us to be good elements in society,*

*And **My friends** who were the great company through the ups and downs, for the beautiful memories for which I'm endlessly grateful,*

*This humble work would have not seen the daylight without **Mme. SETTOUTI N.** and **Mr. BECHAR M.E.A** who greatly contributed to this work, advising me for months, thanks to them for all their efforts, their spirit of collaboration, and their sense of responsibility and commitment towards their duty. Therefore throughout this thesis, I will speak on behalf of us all, as I consider this work not my own but a shared outcome of study and dedication with them,*

*But before all that, I thank **Allah** for everything..*

Acknowledgments

All the work presented in this thesis was done under the supervision of of ***Mme. SETTOUTI N.*** *and* ***Mr. BECHAR M.E.A*** *who have instilled in me the passion for this complex and fascinating subject and helped me navigate it since the beginning to ensure the academic value of this master thesis.*

I also thank my professors for guiding me during my journey in university and inspiring me to reach that point, thanks to ***Mr. Bouizem*** *with whom I refreshed my passion towards mathematics which broaden my vision into the academic research, and* ***Mme. SETTOUTI N.*** *for being such a great dedicated instructor for many years, I thank* ***Mr. BECHAR M.E.A*** *for helping me with the coding and for his suggestions and useful links and recommendations that were crucial to this outcome.*

Special Thanks to the Jury members ***Dr. EL HABIB DAHO Mostafa*** *and* ***Dr. BEHADADA Omar*** *along with my advisers* ***Mme. SETTOUTI N.*** *and* ***Mr. BECHAR M.E.A*** *for taking time reading and reviewing my master thesis.*

Résumé

Les réseaux de neurones convolutifs (CNNs) sont les méthodes de pointe pour l'analyse d'images, ils ont été appliqués à différentes tâches avec une grande variété d'architectures et ils ont réalisé des résultats exceptionnels. Les méthodes basées sur CNN sont généralement une approche à appliquer lorsqu'il s'agit de problèmes complexes avec un jeu de données d'images volumineux, grace à leur capacité à apprendre des représentations profondes et abstraites (deep features) sur l'image. Dans ce projet de fin d'études, nous avons présenté une nouvelle approche basée sur le CNN pour la classification du cancer du sein sur la base de données INbreast pour laquelle nous avons utilisé un VGG-16 pré-entrainé, nous avons implémenté notre méthode avec Python sur le service de cloud computing Google "Colab". Delà, nous avons pu exécuter le modèle VGG-16 et extraire efficacement les bonnes caractéristiques (features). Les résultats obtenus atteignent un taux de classification de 97% avec SVM ce qui est comparable aux méthodes de pointe de classification du cancer du sein, en particulier celles effectuées sur INbreast.

Keywords

Réseaux de neurones convolutifs, caractéristiques profondes et abstraites, VGG-16, Transfer learning, classification, base de données INbreast.

Abstract

Convolutional neural networks (CNNs) are the actual state-of-the-art methods for image analysis, it have been applied to different tasks with a wide range variety of architectures and settings and achieved outstanding results. CNN-based methods are generally such a promising truck to follow when dealing with large image data and complex problems, for their capacity of learning abstract representations of the image which can be used for different image analysis tasks. In this study, we have presented a novel CNN based approach for breast cancer classification on the INbreast dataset, in which we've used a pre-trained and fine-tuned VGG-16 by transfer learning, therefore we've been able to run the training of VGG-16 smoothly and efficiently extracted good features. We have achieved great accuracy of 97% with SVM which can be compared with state-of-the-art methods of breast cancer classification especially those performed on the INbreast dataset.

Keywords

Convolutional neural networks, Deep features, VGG-16, Transfer learning, classification, INbreast database.

Contents

Acknowledgments i
Résumé ii
Abstract iii
Contents iv
List of Figures vi
List of Tables viii
List of source codes ix
Glossary x

Introduction **1**

1 Background **3**
1 Deep Learning 3
1.1 Artificial Neural Network 4
2 Convolutional Neural Network 5
2.1 Biological Inspiration: 5
2.2 CNN Architecture Overview: 7
2.3 The basic CNN architecture: Common layers in CNN 8
2.3.1 Convolutional Layer ($CONV$): 9
2.3.2 Activation Layer (ACT or $ReLU$): 10
2.3.3 Pooling Layer ($POOL$): 11
2.3.4 Fully-Connected Layer (FC): 13
2.4 The Role of Convolutions in Deep Learning: 13
2.5 Visualisation of learned features by CNN: 14
2.6 Pretrained CNNs: 15
2.7 CNNs of Note: 15
3 Transfer Learning 16
4 Computer Vision 19
4.1 Medical image analysis: 21
4.2 Image classification 22
4.3 Image features 23
4.3.1 Classical, hand-crafted features descriptors 25
4.3.2 Advanced, latent-features representations 28
4.3.3 Deep features end-to-end learning methods 28
5 Conclusion 32

2 State of the art **33**
1 Deep features related works on Brain Tumor detection 33
1.1 Comparison 34

2 Deep features related works on Breast cancer recognition 34
2.1 Comparison 35
3 Deep features related works on Lung Nodules segmentation 36
3.1 Comparison 37
4 Deep features related works on Skin Lesions detection 37
4.1 Comparison 38
5 General Synthesis 38
6 Conclusion 40

3 Implementation **41**
1 Introduction: 41
1.1 Starting point: end-to-end Li Shen's study 42
1.2 INbreast dataset 43
1.3 The VGG-16 model 45
1.4 Development environment: Google Colaboratory 46
2 Code implementation 47
2.1 Preparing the environment 47
2.2 Preparing the INbreast dataset: 48
2.3 Setting up the model and the variables for feature extraction 48
2.4 Features extraction: training features 51
2.5 Features extraction: test features 52
2.6 Preparing the labels 53
2.7 Checking features and labels shapes 54
2.8 Classification with SVM 55
2.9 Classification with Multilayer Perceptron 56
2.10 Classification with Random Forest 58
3 Synthesis discussion: 60
3.1 Part 01: Results interpretation 60
3.2 Part 02: Fair Comparison with Li Shen's Study 61
4 Conclusion 63

General conclusion and perspectives **64**

Bibliography **66**

List of Figures

1.1 A) Diagram of the experimental setup of hubel and Wissel experiment showing an extracellular electrode recording from a neuron in the primary visual cortex of a cat. B) In this example, the neuron being recorded from in V1 responds selectively to bars of light presented on the screen in different orientations; the cell fires action potentials (indicated by the vertical lines) only when the bar is at a certain location on the screen and in a certain orientation. These selective responses to stimuli define each neuron's receptive field properties. (After Purves, Augustine, et al., 2008 [9]) [10] 6
1.2 The architecture of the neocognitron. [13] 7
1.3 LeNet-5: One of the earliest convolutional neural networks. [5] . . 7
1.4 High-level general CNN architecture. [5] 8
1.5 Building blocks of a typical CNN. [14] 8
1.6 Convolution layer with input and output volumes. [8] 9
1.7 The convolution operation. [8] 10
1.8 An example of an input volume going through a $ReLU$ activation, $max(0; x)$. Activations are done in-place so there is no need to create a separate output volume although it is easy to visualize the flow of the network in this manner. [16] 11
1.9 **Left:** Our input 4x4 volume. **Right:** Applying 2x2 max pooling with a stride of $S = 1$. **Bottom:** Applying 2x2 max pooling with $S = 2$ – this dramatically reduces the spatial dimensions of our input. [16] 12
1.10 Examples of activation visualizations in different layers based on Zeiler and Fergus's work [18]. Reprinted from [18] with permission. © Springer International Publishing Switzerland, 2014. [5] . . 14
1.11 Different learning processes between (a) traditional machine learning and (b) transfer learning. [23] 17
1.12 Three ways in which transfer might improve learning: a higher performance at the very beginning of learning, a steeper slope in the learning curve, or a higher asymptotic performance [24]. 18
1.13 Some examples of computer vision algorithms and applications. [27] 20
1.14 Images and their histograms. [32] 25
1.15 Low-level feature detection. [38] 26

1.16 Illustration of a deep learning model in which the extracted features in each layer are visualised. Early layers contains simple features (e.g. edges, corners), the deeper the layer is the more abstract the feature gets, means complex representations are a combination of the simple ones. [1] 29

3.1 Converting a patch classifier to an end-to-end trainable whole image classifier using an all convolutional design where they considered removing the heatmap to improve information flow and convolutional layers as top layers; the magnifying glass shows an enlarged version of the heatmap. [58] 42
3.2 Comparison of representative mammograms from DDSM and INbreast. [58] 43
3.3 Database examples: multiple findings. (a) Craniocaudal view of the right breast; (b) mediolateral oblique view of the right breast. [57]. 44
3.4 Charts of (a) the BI-RADS image distribution (b) benign/malignant cases distribution. [57]. 44
3.5 Chart describing the findings in the INbreast database. [57]. 45
3.6 VGG-16 architecture diagram. 46
3.7 Screenshot of Colab environment with dark theme. 47
3.8 Output of code 2: Training and test image data. 48
3.9 Output of code 3: Display of the architecture of the model 50
3.10 Output of code 7: Features and labels shape. 54
3.11 Output of code 8: SVM classification and evaluation. 55
3.12 Output of code 9: Multilayer Perceptron classification and evaluation. 57
3.13 Output of code 10: Random forest classification and evaluation. . . 59
3.14 Confusion matrix for SVM and MLP. 61
3.15 Confusion matrix for RF. 61
3.16 ROC curves for SVM and MLP. 63
3.17 ROC curve for RF. 63

List of Tables

2.1 A summary table of the stated approaches and their full properties. 39

3.1 Classification Results: ACC, Precision, Recall & F1-score. 60
3.2 Classification Results: AUC. 62

List of source codes

1 Import some libraries 47
2 Training and test image data 48
3 Loading the VGG-16 and variables 49
4 Extraction of training features. 51
5 Test features extraction. 52
6 Reading and splitting labels. 53
7 Features and labels shape. 54
8 SVM classification and evaluation. 55
9 Multilayer Perceptron classification and evaluation. 56
10 Random forest classification and evaluation. 58

Glossary

1D: 1 Dimensional.
2D: 2 Dimensional.
3D: 3 Dimensional.
4D: 4 Dimensional.
ACC: Accuracy (overall accuracy).
ACM: Association for Computing Machinery.
ACR: American College of Radiology.
AI: Artificial Intelligence.
AlexNet: Alex Network (referring to Alex Krizhevsk).
ANNs: Artificial Neural Networks.
AUC: Area Under Curve (referring ROC curve).
BI-RADS: Breast Imaging Reporting and Data System.
BN: Batch Normalization.
BRATS: Brain Tumor Segmentation.
BreaKHis: Breast Cancer Histopathological (referring to the BreaKHis dataset).
BRIEF: Binary Robust Independent Elementary Features.
CBIS-DDSM: Curated Breast Imaging Subset of DDSM.
CBR: Case-Based Reasoning.
CC: Color Correlogram.
CC: Craniocaudal(view).
CCV: Color Coherence Vector.
CHSJ: Centro Hospitalar de São João.
CIFAR: Canadian Institute for Advanced Research.
CNN-S: Slow CNN.
CNNs : Convolutional Neural Networks.
Colab: Google Colaboratory.
CONV: Convolutional (referring to the layer).
ConvNet: Convolutional Neural Networks.
CT: Computed Tomography.
CUs: Computing Units.
CV: Computer Vision.
DCNN: Deep Convolutional Neural Network.
DDSM: Digital Database for Screening Mammography.
DenseNet: Densely Connected Convolutional Networks.
DL: Deep Learning.
DO: Dropout.
DREAM2016: referring to the Digital Mammography DREAM 2016 Challenge.
FAST: Features from Accelerated Segment Test. FC: Fully connected (referring to

the layer).
FFDM: Full-Field Digital Mammography.
FN: False Negative.
FP: False Positive.
FT: Features Transfer.
GCM: Gray-level Co-occurrence Matrix.
GLCM: Gray Level Co-occurrence Matrix.
GoogLeNet: Google-LeCun Network.
GPUs: Graphics Processing Units.
Hist.:Histogram.(referring to Histogram information)
HOG: Histogram of Oriented Gradients.
HT: Hough transform.
HUs: Hounsfield units.
IEEE: Institute of Electrical and Electronics Engineers.
ILSVRC: ImageNet Large Scale Visual Recognition Competition.
ImageNet: (referring to ILSVRC database).
ISBI: IEEE International Symposium on Biomedical Imaging.
ISIC: International Skin Imaging Collaboration.
ISLES: Ischemic Stroke Lesion Segmentation (referring to ISLES challenge).
JSRT: Japanese Society of Radiological Technology (referring to the JSRT dataset).
KNN: K-Nearest Neighbors.
KPCA: Kernel Principal Component Analysis.
LBP: Local Binary Pattern.
LG: Logistic Regression.
LIDC-IDRI: Lung Image Database Consortium and Image Database Resource Initiative.
LROIs: Lesion Regions of Interest.
MDS: Multidimensional Scaling.
MIA: Medical Image Analysis.
ML: Machine Learning.
MLO: Mediolateral Oblique (view).
MLP: Multilayer Perceptron.
MRI: Magnetic Resonance Imaging.
NIPS17: Neural Information Processing Systems 2017 conference.
PCA: Principal Component Analysis.
POIs: Points of Interest.
POOL: Pooling (referring to the layer).
ReLU: Rectified Linear Unit (referring to the layer).
ResNet: Residual Neural Network.
RF: Random Forests.
RI: Random Initialization.
RNNs: Recurrent Neural Networks.
ROC: Receiver Operating Characteristic (referring ROC curve).
ROIs: Regions of Interest.
SGF: Steerable Gaussian Filter.
SIFT: Scale invariant feature transform.
SURF: Speeded up robust features.

SVM: Support Vector Machine.
SVC: C-Support Vector Classification.
TL : Transfer Learning.
TN: True Negative.
TP: True Positive.
TPU: Tensor processing unit.
USA: United States of America.
VGG-16: Visual Geometry Group Network-16 (referring to depth of network).
X-Ray: X-Radiography.
XML: Extensible Markup Language.
YaroslavNet: Yaroslav Ganin Network.
ZFNet: Zeiler and Fergus Network.

Introduction

Human Healthcare is an important element if it's not the most important above all concerns, such a field worthy of all complete and effort to improve quality which leads eventually to saving lives, from that point I drew my motivation and inspiration which led me to explore this piece of work!

Modern medicine mostly relies on the data acquired from different imaging modalities that explore the human body interior and dig into lower microscopic levels to better understand the diseases and assist doctors and therapists in their diagnosis and/or treatment decisions.

Since the understanding of the data is such a crucial basis to the whole process, the better we analyze it the more efficient the curing process can be, from here the "Artificial Intelligence" (or AI) more specifically "Machine learning" (or ML) as well as "Computer Vision" (or CV) can help to understand the structural representation of data from a whole different level.

Within the last few decades, AI and CV have empowered their fundamental theories and techniques which has broadened medical challenges to new large perspectives, since there have been introduced many original approaches such as image classification, clustering, image segmentation, etc. that have proven to greatly help to treat and manage many critical situations with considerable accuracy within a short amount of time.

Until recently, the huge advances in both hardware and software areas, the expanding amounts of data, and its open-accessibility have encouraged an emerging research and development outcomes, introducing the "Deep learning" and the "Data Mining" as new concepts, and within comes new revolutionary algorithms and approaches like the one we're interested about in this modest work which is "Convolutional neural networks" (or CNNs) for classification.

In this modest work, we aim to apply one of the recent and the most successful deep learning approaches to enhance the classical concept of classification for medical images, this CNN-based approach is about learning deep features (latent representations) from images with CNN, then using this acquired knowledge to classify a medical image dataset, we chose the INbreast dataset for breast cancer classification.

This thesis is structured mainly in **three chapters**, an **introduction**, and a **conclusion**. The **first chapter** is the **background** chapter in which we necessary theoretical concepts and definitions to acquire a global understanding for the following chapters, the **second chapter** is the **state-of-the-art** chapter in which we review and discuss some notable papers with different approaches all of which have similar aspects and common view with our work to get familiar with the recent developments in the field, the **third and last chapter** is about the **Code implementation** of the approach with Python and discusses the results obtained.

1 Background

In this first chapter, we outline the minimum theoretical basis that is necessary to well assimilate the main matter of the presented work.

As the title of this thesis refers, the approach proposed draws equally from deep learning and computer vision with the special touch of transfer learning, thus we present an overview of each necessary concept in these fields, and explore the most important techniques in brief technical descriptions. Throughout the following sections, we aim to introduce the novice reader to the theoretical background, a common notation, and a general view of each area in this work.

1 Deep Learning

Deep learning is a sub-field of machine learning, which is, in turn, a sub-field of artificial intelligence (AI). It tends to be specifically interested in pattern recognition and learning from data.

It may be surprising to know that the Deep Learning (DL) field has been around since the 1940s [1], undergoing various name changes and incarnations including cybernetics, connectionism, and the most familiar, Artificial Neural Networks (or ANNs), based on research trends, available hardware and datasets, and popular options of prominent researchers at the time.

> — Deep learning methods are representation-learning methods with multiple levels of representation, obtained by composing simple but non-linear modules that each transform the representation at one level (starting with the raw input) into a representation at a higher, slightly more abstract level. [...] The key aspect of deep learning is that these layers are not designed by human engineers: they are learned from data using a general-purpose learning procedure
>
> *Yann LeCun, Yoshua Bengio, and Geoffrey Hinton, Nature 2015. [2]*

While other approaches to ML tend to focus on learning only one or two layers of representations of the data; hence, they're sometimes called shallow learning.

DL on the other hand is a new take on learning representations from data that emphasizes learning successive layers of increasingly meaningful representations. The deep in DL isn't a reference to any kind of deeper understanding achieved by the approach; rather, it stands for this idea of successive layers of representations. The number of layers that contribute to a model of the data is called the depth of the model. Modern deep learning often involves tens or even hundreds of successive layers of representations and they've all learned automatically from exposure to training data [3]. These representations are often learned via models called neural networks, structured in literal layers stacked on top of each other.

Deep learning enables data-driven decisions by identifying and extracting patterns from large datasets that accurately map from sets of complex inputs to good decision outcomes. In the healthcare sector, deep learning is used to process medical images (X-rays, CT, and MRI scans) and diagnose health conditions [4].

1.1 Artificial Neural Network

ANNs are popular ML techniques that simulate the mechanism of learning from biological organisms inspired by the structure and function of the brain. It learns from data and specializes in pattern recognition.

While inspired by the human brain and how its neurons interact with each other, ANNs are not meant to be realistic models of the brain, instead, they are an inspiration, allowing us to draw parallels between a very basic model of the brain and how we can mimic some of this behavior. Deep learning belongs to the family of ANN algorithms, and in most cases, the two terms can be used interchangeably.

An Artificial neural network contains computation units that are referred to as neurons. These CUs are connected through weights, which serve the same role as the strengths of synaptic connections in biological organisms. Each input to a neuron is scaled with a weight, which affects the function computed at that unit. An ANN computes a function of the inputs by propagating the computed values from the input neurons to the output neuron(s) and using the weights as intermediate parameters.

Learning occurs by changing the weights connecting the neurons. Just as external stimuli are needed for learning in biological organisms, the external stimulus in ANNs is provided by the training data containing examples of input-output pairs of the function to be learned.

The weights between neurons are adjusted in a neural network in response to prediction errors. The goal of changing the weights is to modify the computed function to make the predictions more correct in future iterations.

From this point of view, an ANN can be viewed as a computational graph of elementary units in which greater power is gained by connecting them in particular ways. Furthermore, sufficient training data is also required to learn the larger number of weights in these expanded computational graphs. [5]

2 Convolutional Neural Network

Convolutional networks (LeCun, 1989) [6], also known as convolutional neural networks, or CNNs, are a specialized kind of neural network for processing data that has a known grid-like topology. Examples include time-series data, which can be thought of as a 1-D grid taking samples at regular time intervals, and image data, which can be thought of as a 2-D grid of pixels. Convolutional networks have been tremendously successful in practical applications. The name "convolutional neural network" indicates that the network employs a mathematical operation called convolution. Convolution is a specialized kind of linear operation. Convolutional networks are simply neural networks that use convolution in place of general matrix multiplication in at least one of their layers. [1]

2.1 Biological Inspiration:

The biological inspiration for CNNs is the visual cortex in animals [7]. The cells in the visual cortex are sensitive to small subregions of the input. We call this the visual field (or receptive field). These smaller subregions are tiled together to cover the entire visual field. The cells are well suited to exploit the strong spatially local correlation found in the types of images our brains process, and act as local filters over the input space. There are two classes of cells in this region of the brain. The simple cells activate when they detect edge-like patterns, and the more complex cells activate when they have a larger receptive field and are invariant to the position of the pattern. [8]

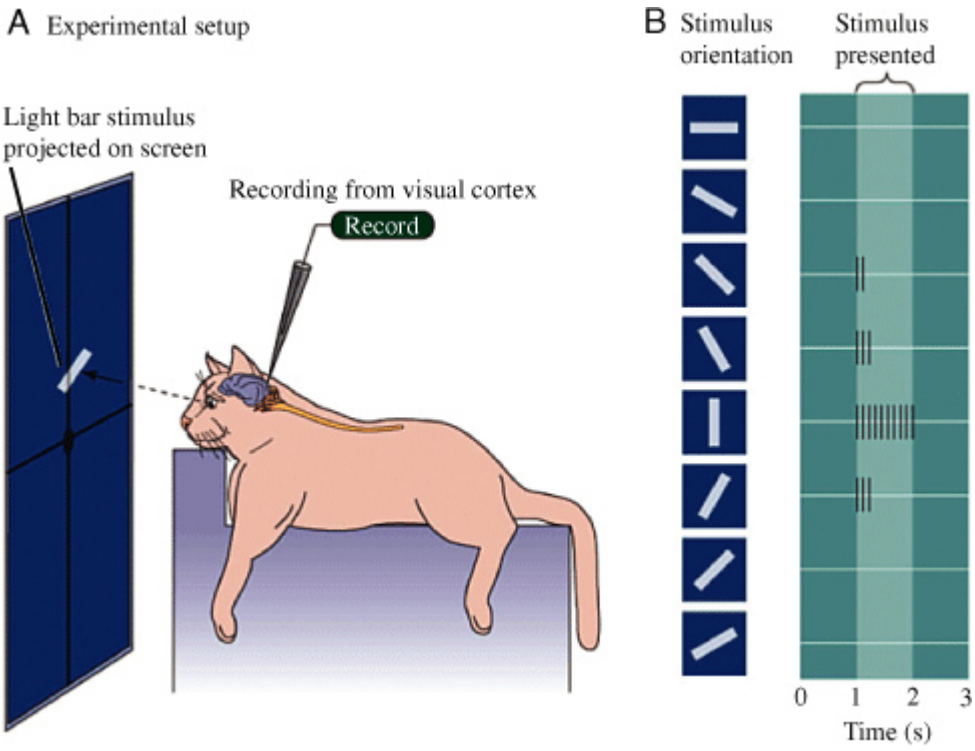


Figure 1.1: A) Diagram of the experimental setup of hubel and Wissel experiment showing an extracellular electrode recording from a neuron in the primary visual cortex of a cat. B) In this example, the neuron being recorded from in V1 responds selectively to bars of light presented on the screen in different orientations; the cell fires action potentials (indicated by the vertical lines) only when the bar is at a certain location on the screen and in a certain orientation. These selective responses to stimuli define each neuron's receptive field properties. (After Purves, Augustine, et al., 2008 [9]) [10]

This broader principle was used to design a sparse architecture for convolutional neural networks. The first basic architecture based on this biological inspiration was the neocognitron, which was then generalized to the LeNet-5 architecture [11]. [12]

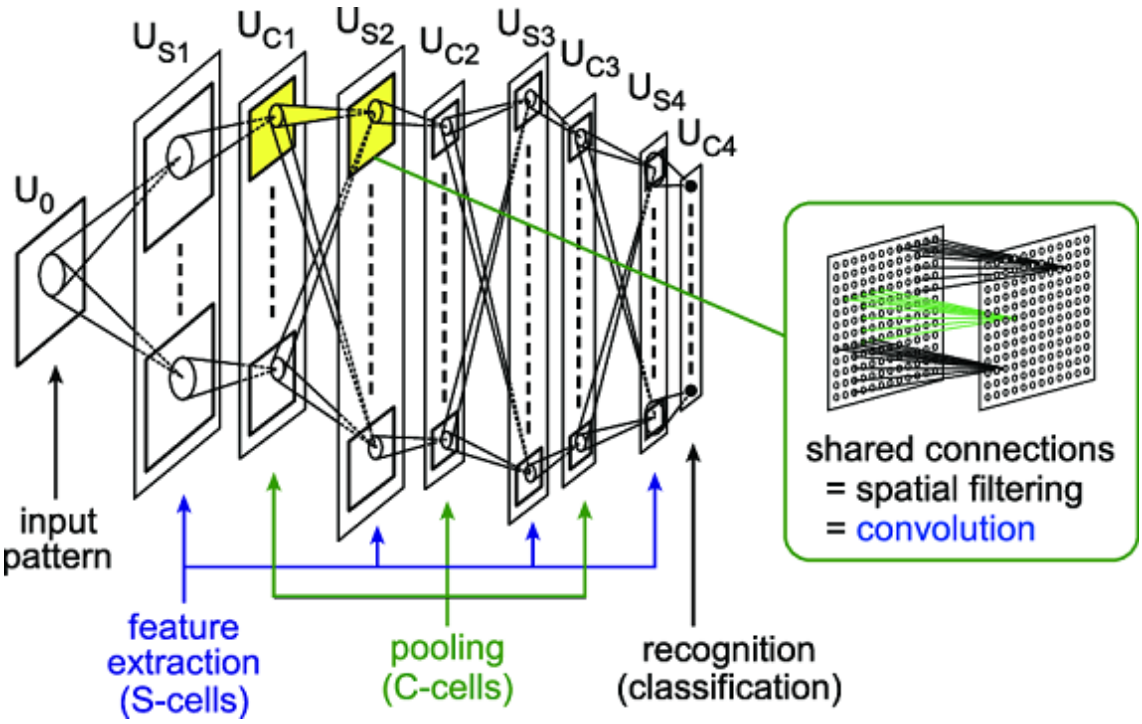


Figure 1.2: The architecture of the neocognitron. [13]

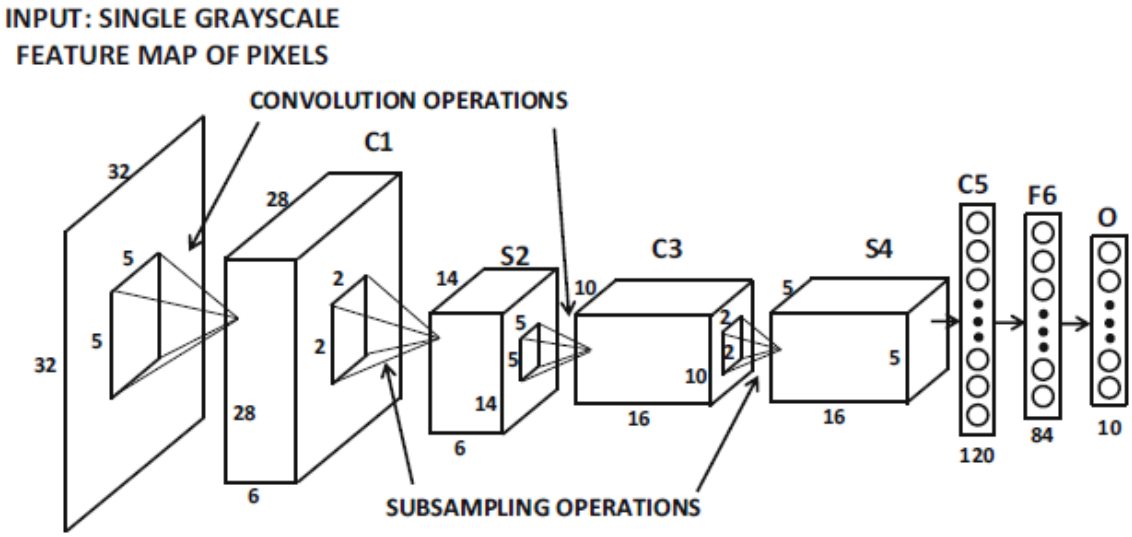


Figure 1.3: LeNet-5: One of the earliest convolutional neural networks. [5]

2.2 CNN Architecture Overview:

CNN transform the input data from the input layer through all connected layers into a set of class scores given by the output layer. There are many variations of the CNN architecture, but they are based on the pattern of layers, as demonstrated in Figure 1.4. [8]

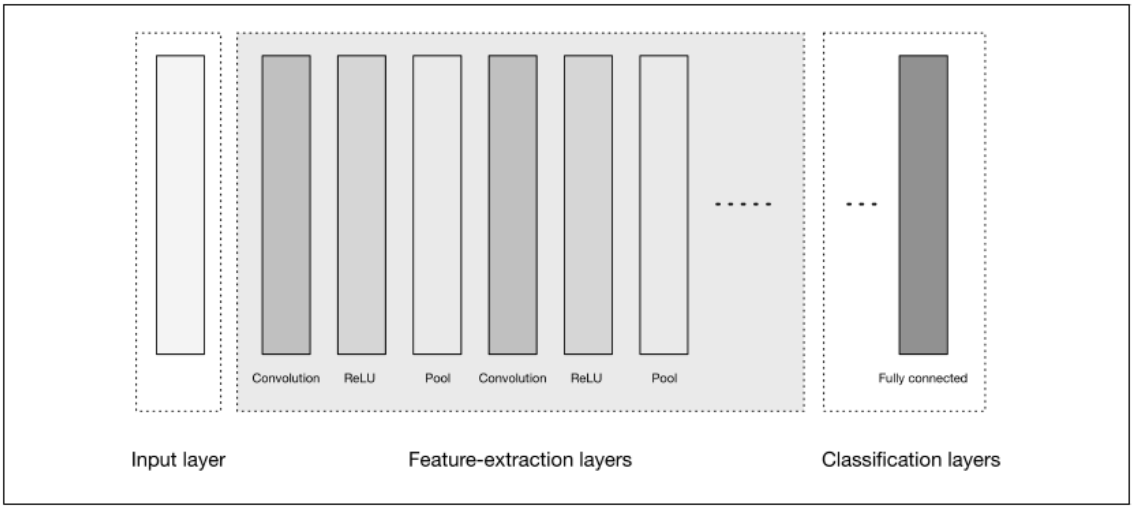


Figure 1.4: High-level general CNN architecture. [5]

2.3 The basic CNN architecture: Common layers in CNN

In CNN, the states in each layer are arranged according to a spatial grid structure. These spatial relationships are inherited from one layer to the next because each feature value is based on a small local spatial region in the previous layer.
Each layer in the CNN is a 3-dimensional grid structure, which has height, width, and depth. The depth of a layer should not be confused with the depth of the network itself. [5]

The word "depth" (of a single layer) refers to the number of channels in each layer, such as the number of primary color channels (e.g., blue, green, and red) in the input image or the number of feature maps in the hidden layers while the depth (of the whole network) is the number of layers of the model itself.

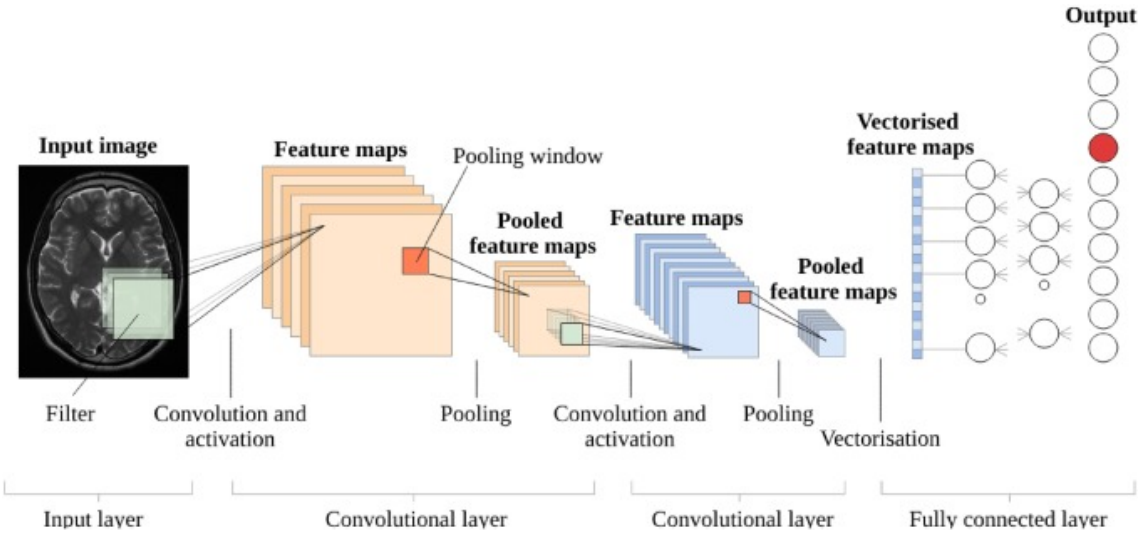


Figure 1.5: Building blocks of a typical CNN. [14]

The CNN functions much like a traditional feed-forward neural network, except that the operations in its layers are spatially organized with sparse (and carefully designed) connections between layers.
The three types of layers that are commonly present in a CNN are convolution, pooling, and ReLU. [5] (See figure 1.5).

2.3.1 Convolutional Layer ($CONV$):

Convolutional layers are considered the core building blocks of CNN architectures. As Figure 1.6 illustrates, convolutional layers transform the input data by using a patch of locally connecting neurons from the previous layer. The layer will compute a dot product between the region of the neurons in the input layer and the weights to which they are locally connected in the output layer. [8]

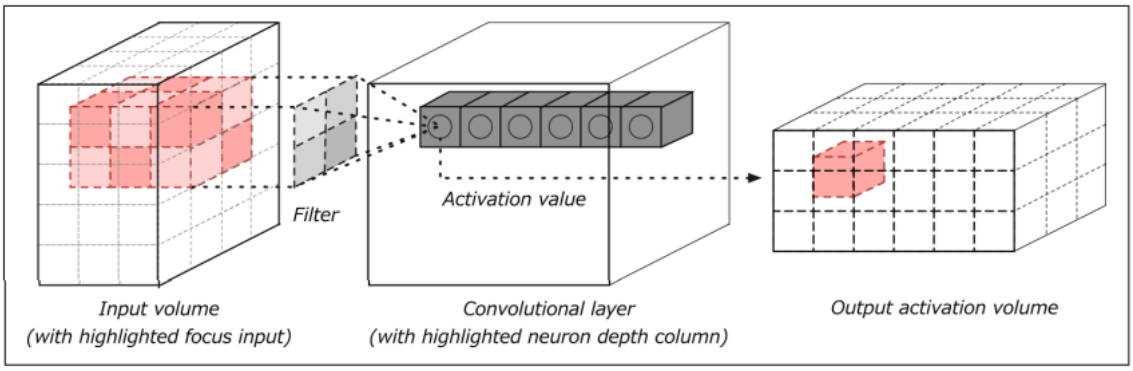


Figure 1.6: Convolution layer with input and output volumes. [8]

In convolutional network terminology, the first argument (in this example, the function x) to the convolution is often referred to as the input and the second argument (in this example, the function w) as the kernel. The output is sometimes referred to as the feature map. [1]

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a) \tag{1.1}$$

We commonly refer to the sets of weights in a convolutional layer as a filter (or kernel). This filter is convolved with the input and the result is a feature map (or activation map). Convolutional layers perform transformations on the input data volume that are a function of the activations in the input volume and the parameters (weights and biases of the neurons). The activation map for each filter is stacked together along the depth dimension to construct the 3D output volume. [8]

Convolution A convolution is defined as a mathematical operation describing a rule for how to merge two sets of information.

The convolution operation, shown in Figure 1.7, is known as the feature detector of a CNN. The input to a convolution can be raw data or a feature map output from another convolution. It is often interpreted as a filter in which the kernel filters input data for certain kinds of information; for example, an edge kernel lets pass through only information from the edge of an image. [8]

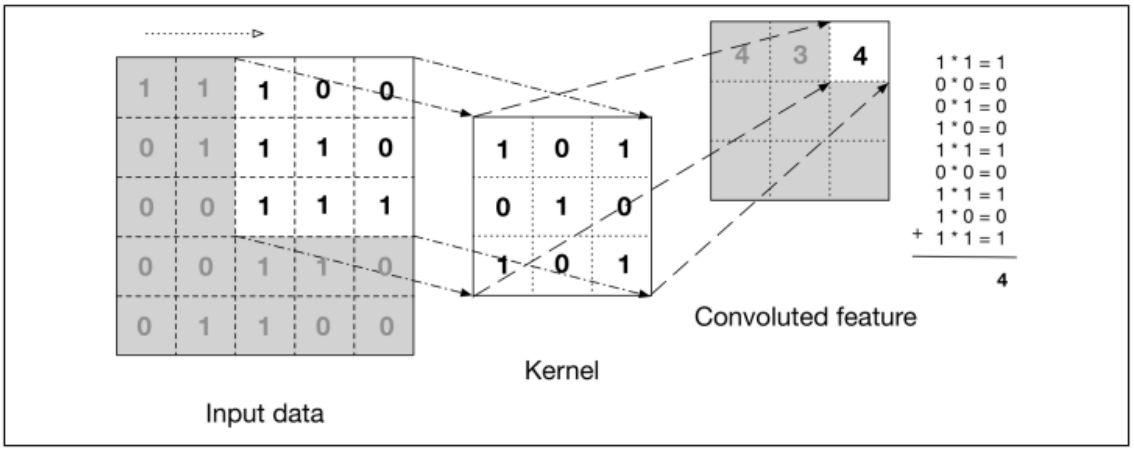


Figure 1.7: The convolution operation. [8]

2.3.2 Activation Layer (ACT or $ReLU$):

An activation function is a mathematical function that transforms the input data from the previous layer into a meaningful representation, which is closer to the expected output. [15]

Activation Functions can be: Linear Activation Function or Nonlinear Activation Functions.

After each $CONV$ layer in a CNN, we apply a non-linear activation function, for which the most commonly used is the $ReLU$ (Rectified Linear Units). We typically denote activation layers as ACT or simply $ReLU$.

Activation layers are not technically "layers" (since no parameters/weights are learned inside an activation layer) and are sometimes omitted from network architecture diagrams as it's assumed that an activation immediately follows a convolution. [16]

As an example, consider the following network architecture:

$$INPUT \Rightarrow CONV \Rightarrow ReLU \Rightarrow FC$$

To make this diagram more concise, we could simply remove the $ReLU$ component since it's assumed that an activation always follows a convolution:

$$INPUT \Rightarrow CONV \Rightarrow FC$$

An activation layer accepts an input volume of size $W_{\text{input}} \times H_{\text{input}} \times D_{\text{input}}$ and then applies the given activation function (Figure 1.8). [16]

With CNNs, we often see $ReLU$ layers used. The $ReLU$ layer will apply an elementwise activation function over the input data thresholding—for example, max(0,x)—at zero, giving us the same dimension output as the input to the layer: [8]

$$W_{\text{input}} = W_{\text{output}}, H_{\text{input}} = H_{\text{output}}, D_{\text{input}} = D_{\text{output}}$$

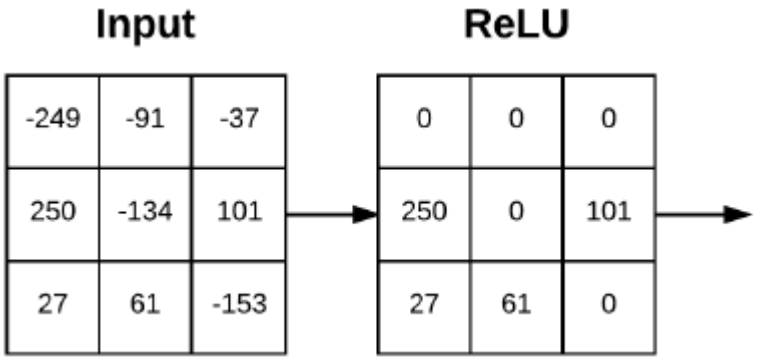


Figure 1.8: An example of an input volume going through a $ReLU$ activation, $max(0; x)$. Activations are done in-place so there is no need to create a separate output volume although it is easy to visualize the flow of the network in this manner. [16]

It is noteworthy that the use of the $ReLU$ activation function is a recent evolution in neural network design. In the earlier years, saturating activation functions like sigmoid and tanh were used. However, it was shown in [17] that the use of the $ReLU$ has tremendous advantages over these activation functions both in terms of speed and accuracy. Increased speed is also connected to accuracy because it allows one to use deeper models and train them for a longer time. [5]

Rectified linear units (ReLU) are the current state of the art because they have proven to work in many different situations. Because the gradient of a ReLU is either zero or a constant, it is possible to reign in the vanishing exploding gradient issue. ReLU activation functions have been shown to train better in practice than sigmoid activation functions. [8]

In recent years, the use of the $ReLU$ activation function has replaced the other activation functions in CNN design. [5]

2.3.3 Pooling Layer ($POOL$):

Pooling layers are commonly inserted between successive convolutional layers. We want to follow convolutional layers with pooling layers to progressively reduce the spatial size (width and height) of the data representation.

Pooling layers reduce the data representation progressively over the network and help control overfitting. The pooling layer operates independently on every depth slice of the input. [8]

It is common to insert POOL layers in-between consecutive CONV layers in a CNN architectures: [16]

$$INPUT \Rightarrow CONV \Rightarrow RELU \Rightarrow POOL \Rightarrow CONV \Rightarrow RELU \Rightarrow POOL \Rightarrow FC$$

$POOL$ layers operate on each of the depth slices of an input independently using either the max or average function. Max pooling is typically done in the middle of the CNN architecture to reduce spatial size, whereas average pooling is normally used as the final layer of the network (e.x., GoogLeNet, SqueezeNet, ResNet) where we wish to avoid using FC layers entirely. The most common type of $POOL$ layer is max pooling, although this trend is changing with the introduction of more exotic micro-architectures. [16]

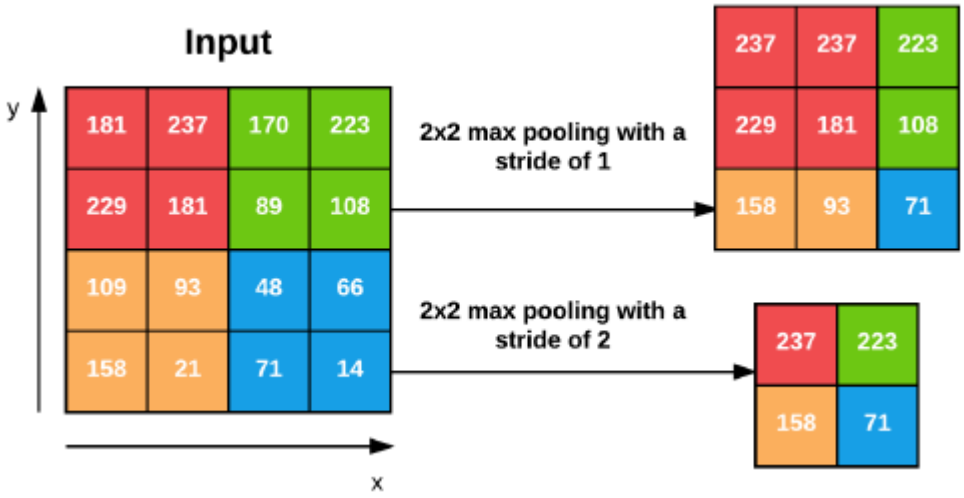


Figure 1.9: **Left:** Our input 4x4 volume. **Right:** Applying 2x2 max pooling with a stride of $S = 1$. **Bottom:** Applying 2x2 max pooling with $S = 2$ – this dramatically reduces the spatial dimensions of our input. [16]

In summary, $POOL$ layers Accept an input volume of size $W_{\text{input}} \times H_{\text{input}} \times D_{\text{input}}$. They then require two parameters: [8]

- The receptive field size F (also called the "pool size")
- The stride S[1]

Applying the $POOL$ operation yields an output volume of size $W_{\text{output}} \times H_{\text{output}} \times D_{\text{output}}$, where: [16]

$$\begin{aligned} W_{\text{output}} &= \left(\left(W_{\text{input}} - F\right)/S\right) + 1 \\ H_{\text{output}} &= \left(\left(H_{\text{input}} - F\right)/S\right) + 1 \\ D_{\text{output}} &= D_{\text{input}} \end{aligned}$$

[1]Stride configures how far our sliding filter window will move per application of the filter function.

2.3.4 Fully-Connected Layer (FC):

Neurons in FC layers are fully-connected to all activations in the previous layer, as is the standard for feedforward neural networks. FC layers are always placed at the end of the network (i.e., we don't apply a CONV layer, then an FC layer, followed by another CONV) layer. [16]

Fully connected layers perform transformations on the input data volume that are a function of the activations in the input volume and the parameters (weights and biases of the neurons). [8]

It's common to use one or two FC layers prior to applying the softmax classifier, as the following (simplified) architecture demonstrates:

$$INPUT \Rightarrow CONV \Rightarrow RELU \Rightarrow POOL \Rightarrow CONV \Rightarrow RELU \Rightarrow POOL \Rightarrow FC \Rightarrow FC$$

Here we apply two fully-connected layers before our (implied) softmax classifier which will compute our final output probabilities for each class. [16]

Multiple Fully Connected Layers: Some CNN architectures will use multiple fully connected layers at the end of the network. AlexNet is an example of this; it has two fully connected layers followed by a softmax layer at the end. [8]

There are other layers for a specific use-case: Batch normalization (BN), Dropout (DO) .etc

2.4 The Role of Convolutions in Deep Learning:

By applying convolutions filters, nonlinear activation functions, pooling, and backpropagation, CNNs are able to learn filters that can detect edges and blob-like structures in lower-level layers of the network – and then use the edges and structures as "building blocks", eventually detecting high-level objects (e.x., faces, cats, dogs, cups, etc.) in the deeper layers of the network.

In the context of image classification, our CNN may learn to:

- Detect edges from raw pixel data in the first layer
- Use these edges to detect shapes (i.e., "blobs") in the second layer
- Use these shapes to detect higher-level features such as facial structures, parts of a car, etc. in the highest layers of the network

The last layer in a CNN uses these higher-level features to make predictions regarding the contents of the image. [16]

2.5 Visualisation of learned features by CNN:

Another excellent set of visualizations from [18] is shown in 1.10: [5]

Figure 1.10: Examples of activation visualizations in different layers based on Zeiler and Fergus's work [18]. Reprinted from [18] with permission. © Springer International Publishing Switzerland, 2014. [5]

2.6 Pretrained CNNs:

Pretrained convolutional neural networks from publicly available resources like ImageNet are often available for use in an off-the-shelf manner for other applications and data sets. This is achieved by using most of the pre-trained weights in the convolutional network without any change except for the final classification layer. The weights of the final classification layer are learned from the data set at hand. The training of the final layer is necessary because the class labels in a particular setting may be different from those of ImageNet. Nevertheless, the weights in the early layers are still useful because they learn various types of shapes in the images that can be useful for virtually any type of classification application.
It is noteworthy that the use of pre-trained convolutional networks is so popular that training is rarely started from scratch. [5]

CNNs as one of the common neural architectures, it's among the earliest success stories of the power of neural networks, also represent excellent examples of how biologically inspired neural networks can sometimes provide ground-breaking results.

Convolutional neural networks have historically been the most successful of all types of neural networks. They are used widely for image recognition, object detection/localization, and even text processing. The performance of these networks has recently exceeded that of humans in the problem of image classification [19].
Convolutional neural networks provide a very good example of the fact that architectural design choices in a neural network should be performed with semantic insight about the data domain at hand. In the particular case of the convolutional neural network, this insight was obtained by observing the biological workings of a cat's visual cortex, and heavily using the spatial relationships among pixels. This fact also provides some evidence that a further understanding of neuroscience might also be helpful for the development of methods in artificial intelligence. [5]

2.7 CNNs of Note:

Following is a list of some of the more popular architectures of CNNs. [8]

- LeNet [11]
  - One of the earliest successful architectures of CNNs
  - Developed by Yann Lecun
  - Originally used to read digits in images

- AlexNet [17]
  - Helped popularize CNNs in computer vision
  - Developed by Alex Krizhevsky, Ilya Sutskever, and Geoff Hinton
  - Won the ILSVRC 2012
- ZF Net [18]
  - Won the ILSVRC 2013
  - Developed by Matthew Zeiler and Rob Fergus
  - Introduced the visualization concept of the Deconvolutional Network
- GoogLeNet [20]
  - Won the ILSVRC 2014
  - Developed by Christian Szegedy and his team at Google
  - Codenamed "Inception", one variation has 22 layers
- VGGNet [21]
  - Runner-Up in the ILSVRC 2014
  - Developed by Karen Simonyan and Andrew Zisserman
  - Showed that depth of network was a critical factor in good performance
- ResNet [19]
  - Trained on very deep networks (up to 1,200 layers)
  - Won first in the ILSVRC 2015 classification task

3 Transfer Learning

Like AI in general and ML in particular, the concept of transfer learning has gone through decades of evolution. From AI's early years, researchers have considered the ability to transfer one's knowledge as one of the fundamental cornerstones of intelligence [22].

In early stages, Transfer Learning (abbreviated TL) has been studied under different terminologies in AI including "*learning by analogy*", "*case-based reasoning (CBR)*", "*knowledge reuse and re-engineering*", "*lifelong machine learning*", "*never-ending learning*", "*fine-tuning*" and "*domain adaptation*" and so on.

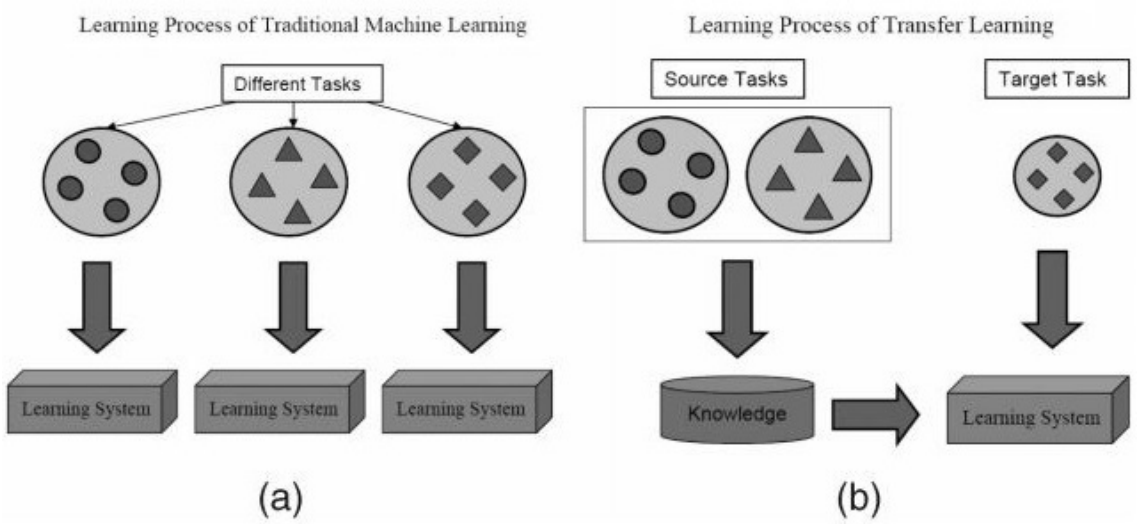


Figure 1.11: Different learning processes between (a) traditional machine learning and (b) transfer learning. [23]

In a nutshell, TL refers to the machine learning paradigm in which an algorithm extracts knowledge from one or more application scenarios to help boost the learning performance in a "*target scenario*". Compared to traditional machine learning, which requires large amounts of well-defined training data as the input, transfer learning can be understood as a new learning paradigm (see figure 1.11).

Definition (transfer learning) Given a source domain $\mathbb{D}_s$ and learning task $\mathbb{T}_s$, a target domain $\mathbb{D}_t$ and learning task $\mathbb{T}_t$, transfer learning aims to help improve the learning of the target predictive function $f_t(\cdot)$ for the target domain using the knowledge in $\mathbb{D}_s$ and $\mathbb{T}_s$, where $\mathbb{D}_s \neq \mathbb{D}_t$ or $\mathbb{T}_s \neq \mathbb{T}_t$. [23]

Based on the transfer learning methodologies, once we obtain a well-developed model in one domain, we can bring this model to benefit other similar domains. Hence, having an accurate "*distance*" measure between any task domains is necessary in developing a sound transfer learning methodology. If the distance between two domains is large, then we may not wish to apply transfer learning as the learning might turn out to produce a negative effect. On the other hand, if two domains are "*close by*," transfer learning can be fruitfully applied (see figure 1.12).

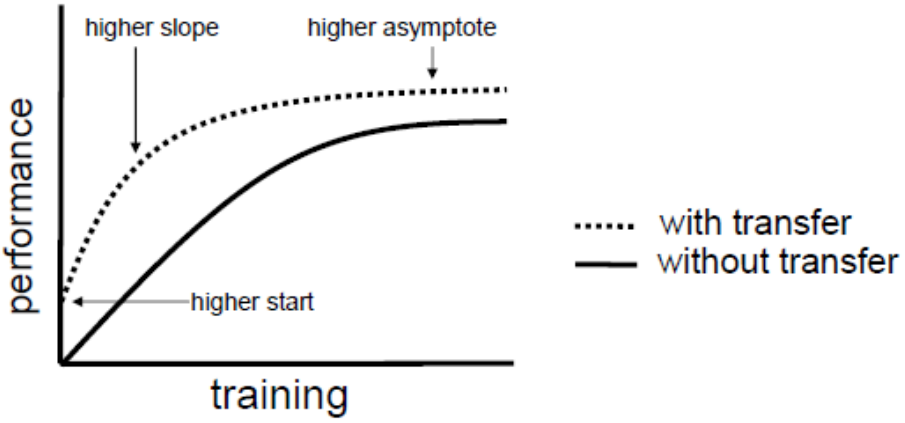


Figure 1.12: Three ways in which transfer might improve learning: a higher performance at the very beginning of learning, a steeper slope in the learning curve, or a higher asymptotic performance [24].

In machine learning, the distance between domains can often be measured in terms of the features that are used to describe the data. In image analysis, features can be pixels or patches in an image pattern, such as the color or shape [22].

Transfer learning becomes particularly important for ML, it demonstrates how learning systems can quickly adapt themselves to new situations, new tasks, and new environments without much effort or cost. Now, such an evolution can be examined in three points:

1. **The ability to learn from small data**: This ability to learn from small data can be partly explained by the ability of humans to leverage and adapt like it's the case of babies who learn from few examples and generalize quickly and effectively to concepts. Adaptation is an innate ability of intelligent beings and intelligent systems should certainly be endowed with transfer learning ability.

2. **Small-sized data sets**: In ML practice, we are often surrounded by lots of isolated and fragmented small-sized data sets, and that's an overwhelming amount of big data which not all organizations can collect or the right to access for some resource limitations constraints and user privacy concerns. This small-data challenge is a serious problem that can be miraculously solved by "*Transfer learning*".

3. **Reliability**: When building a machine learning model, we can not foresee all future situations, this problem is often addressed using a technique known as regularization. Transfer learning takes this approach further, by allowing the model to be complex while being prepared for changes when they come, therefore TL can make AI and ML systems more reliable and robust when the external environment changes, that's said from a software system's perspective.

If we continuously apply transfer learning in our ML practice, we can obtain a lifelong machine learning system that can draw knowledge from a succession of problem-solving experiences, both in a long period and from a large variety of tasks [22].

4 Computer Vision

> — Vision allows humans to perceive and understand the world surrounding them, while computer vision aims to duplicate the effect of human vision by electronically perceiving and understanding an image.
>
> *M. Sonka, R. Boyle and V. Hlavac, "Image Processing, Analysis, and Machine Vision", 2014.[25]*

As "AI" had the ambition to endow machines with some sort of intelligence and automation, and answer the original question of Alan Turing "Can machines think?", "Computer vision" followed AI steps raising a new ambitious question "*can machine see?*" which gave birth to a new revolutionary perspective in the human evolution.

Computer Vision (abbreviated CV), is a multidisciplinary field that could broadly be called a sub-field of AI and ML, which may involve the use of specialized methods and make use of general learning algorithms.
CV is a field of study and research that aim to give computers the ability to observe and interpret the 3D state of the visual world.

CV started in the early 1970s, viewed as the visual perception component of the next intelligent systems. What distinguished "computer vision" from the already existing field of "digital image processing" (which lay only on 2D images) was a desire to recover the three-dimensional structure of the world from images and to use this as a stepping stone towards full scene understanding.

In vision problems, we take visual data x and use them to infer the state of the world w. The world state w may be continuous (the 3 D pose of a body model) or discrete (the presence or absence of a particular object). When the state is continuous, we call this inference process regression. When the state is discrete, we call it classification. [26]

The CV field has developed rapidly but it's not stopping any sooner, as with the emergence of machine learning and especially neural network architectures "computer vision" is exploring cutting-edge frameworks that led to huge advancements like: Data augmentation, feature-based analysis, semantic segmentation, etc.

Two main reasons are behind the rapid progress of CV, one is that the processing power, memory, and storage capacity of computers has vastly increased, the

other reason is as we mentioned before the contribution of ML algorithms which are deployed widely in vision applications.

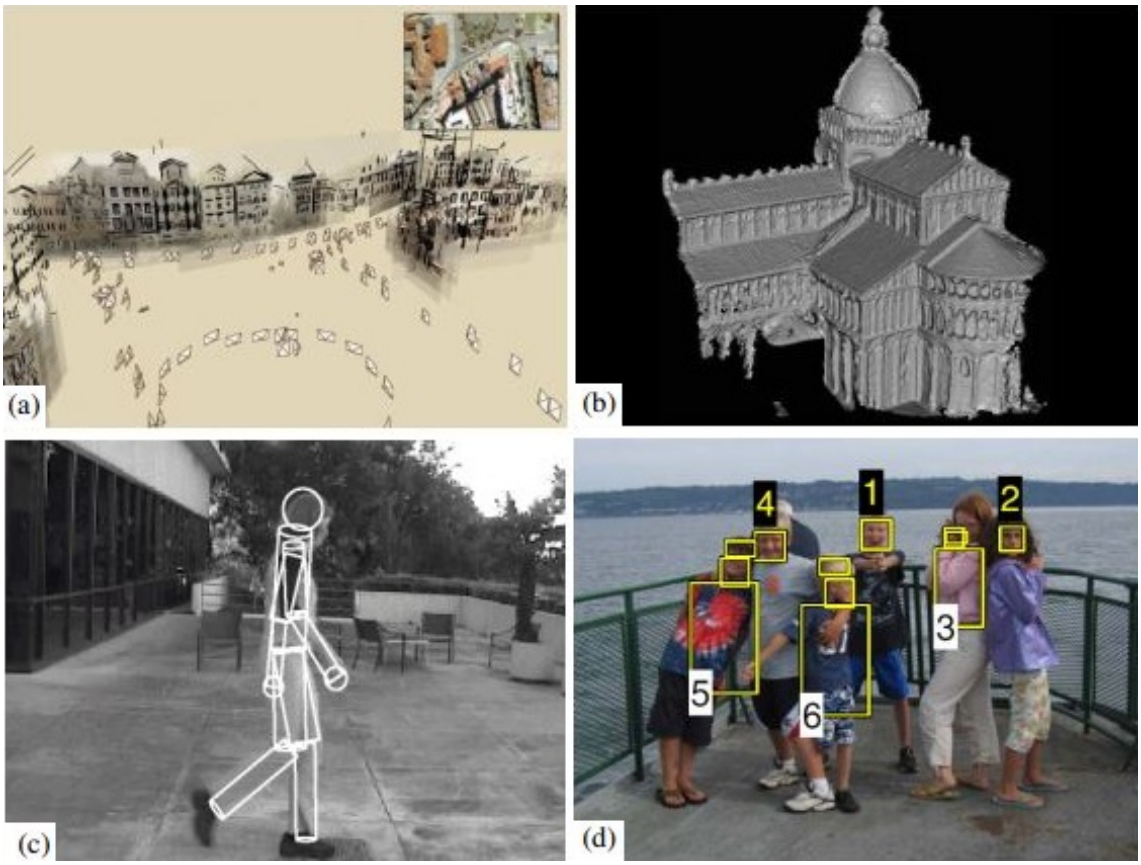


Figure 1.13: Some examples of computer vision algorithms and applications. [27]

CV is being used today in a wide variety of real-world applications. We only mention a few:

- Structure from motion algorithms can reconstruct a sparse 3D point model of a large complex scene from hundreds of partially overlapping photographs (Snavely, Seitz, and Szeliski 2006 [28]) © 2006 ACM. (see figure 1.13.a)
- Stereo matching algorithms can build a detailed 3D model of a building facade from hundreds of differently exposed photographs taken from the Internet (Goesele, Snavely, Curless et al. 2007 [29]) © 2007 IEEE. (see figure 1.13.b)
- Person tracking algorithms can track a person walking in front of a cluttered background (Sidenbladh, Black, and Fleet 2000 [30]) © 2000 Springer. (see figure 1.13.c)
- Face detection algorithms, coupled with color-based clothing and hair detection algorithms, can locate and recognize the individuals in this image (Sivic, Zitnick, and Szeliski 2006 [31]) © 2006 Springer. (see figure 1.13.d) [27]

> — However, despite all of these advances, the dream of having a computer interpret an image at the same level as a two-year-old (for example, counting all of the animals in a picture) remains elusive.
>
> *Richard Szeliski, "Computer vision: algorithms and applications", 2010. [27]*

4.1 Medical image analysis:

Image analysis is the task of extracting abstract information or semantics and knowledge from the raw pixels of image and signal data.
This is the most challenging task in biomedical imaging as it supports researchers and clinicians in finding clues for disease or certain phenotypes (diagnostics), supports novices and experts in performing procedures (therapy) and follow-up to the outcome, and allows scientists to gain knowledge from imaging data. [32]

Medical image analysis (or MIA): is an application of CV, that emerged in the 1990s to automatically complete tasks such as classification, detection, and segmentation for clinical care and biomedical research.
Although MIA is a subfield of computer vision, compared to images in general, medical images have some specific characteristics: [22]

- **Small data and expensive labeling:**
  Medical image data are collected through special equipment under very private contexts, hence, it often only have a small sample, measured in the order of hundreds of samples only.
  The labeling of medical images often relies on experienced and well-trained human experts such as doctors and radiologists, making the labeling of medical images a much more expensive and delicate process.

- **Complex data:**
  CV tasks usually focus on two-dimensional (2D) images or videos. However, medical images have a much more complex data formation, X-ray images often consist of several views of a body area of a patient, a CT examination offers three-dimensional (3D) images or videos, MRI images even have several modalities in 3D images, and Ultrasound devices often generate sequential image data.

- **Imbalanced labels:**
  In practice, medical image data are imbalanced. Positive results such as true confirmations of cancer cases have a much lower chance of appearing than negative results since most patients are healthy. This makes the data have an imbalanced distribution, which in turn makes it difficult to learn.
  Additionally, the abnormality of the data usually occurs in a small patch of the images, and these local patches will determine the label of the whole image. For example, a small tumor in a CT scan of a lung will lead to a positive label no matter what other patches are classified. This is a typical case of multi-instance learning (Dietterich et al., 1997 [33]).

Methodology: A common biomedical image analysis task can be split up into several sub-steps: [32]

- Preprocessing to remove background noise or enhance the image,
- Extraction of features to be used in later steps,
- Registration of several images,
- Segmentation (localization and delineation) of regions of interest (ROIs),
- Classification of the image or segmented parts and measurements.

4.2 Image classification

> — Given a set of training data points, each of which is associated with a class label, determine the class label of one or more previously unseen test instances.
>
> *Charu C. Aggarwal, "Data Mining: The Textbook", 2015. [12]*

As we stated earlier, image classification is an application of CV in a continuous state of the world which can simply refer to the external environment from which we acquire the data.

Image Classification, at its very core, is the task of assigning a label to an image from a predefined set of categories.
Practically, this means that our task is to analyze an input dataset of images and return a label that categorizes each image. The label is always from a predefined set of possible categories [16] (also classes or targets), as the system could also assign multiple labels to the same image via probabilities of how likely the image belongs to each class.

Image Classification Approach can be described by the following steps: [34]

I **Digital data:**
An image is captured by using a digital camera or any other digital instrument.

II **Pre-processing:**
Different operations of improvement on the image data, which includes normalization, contrast enhancement, obtaining a gray-scale image, binary image, resizing image, complementing binary image, removing noise, getting the image boundary.

III **Feature extraction:**
The process of measuring, calculating or detecting the characteristics (or features) from the image samples. The two most common methods are:

i geometric feature extraction
ii color feature extraction.

IV **Selection of preparing information:**
Selection of the specific property which best portrays the given example, e.g., info image, output image, after preprocessing train dataset name.

V **Decision and classification:**
Categorizes recognized items into predefined classes by utilizing a reasonable strategy that contrasts the image designs and the objective examples.

VI **Accuracy evaluation:**
Assesses the performance, precision appraisal is acknowledged to distinguish conceivable wellsprings of mistakes and as a pointer utilized as a part of correlations.

4.3 Image features

Many information processing tasks can be very easy or very difficult depending on how the information is represented. This is a general principle applicable to daily life, computer science in general, and machine learning. [1]

Feature Engineering interests in formulating the most appropriate features and it is considered the essential preamble in any successful ML pipeline.

Generally talking, a feature is a numeric representation of raw data, that describes those defining characteristics of a given dataset that allow for optimal learning.
A feature is designed to be informative to the system when processing it, therefore a set of accurate features seek to explain an entire phenomenon.
Features are a way of representing or coding and passing abstract human understanding to computers. There are many ways to forge features, which is why features can have many forms (structured or unstructured, quantitative or qualitative, etc.).

However, broadly speaking the quality of features we can design is fundamentally dependent on our level of knowledge regarding the phenomenon we are studying. The more we understand (both intellectually and intuitively) the process of generating the data we have at our fingertips, the better we can design features ourselves or, ideally, teach the computer to do this design work itself.
Indeed, well-designed features are crucial to the performance of both regression and classification schemes. [35]

Indeed, the performance of many machine learning algorithms heavily depends on having insightful input representations that expose the underlying explanatory factors of the output for the observed input [36].

An effective data representation would also reduce data redundancy and adapt to undesired, external factors of variation introduced by sensor noise, labeling errors, missing samples, etc.
In the case of images or videos, dimensionality reduction is often an integral part of feature engineering, since the raw data are typically high dimensional. [37]

Most representation learning problems face a trade-off between preserving as much information about the input as possible and attaining nice properties (such as independence). [1]

Effective feature engineering tools and feature extraction methods are one of the main research areas of ML, which are still far from being solved adequately. However, throughout the short history of ML, we've known many approaches that have proven their success at a certain period and new perspectives involving deep learning are yet to be tuned.

> — If computer vision was an eye to a system, feature extraction would be its retina.

In its broadest sense, an image is a spatial map of one or more physical properties of a subject where the pixel intensity represents the value of a physical property of the subject at that point. [32]
Image features are simplified descriptors of some aspect of an image, part of it, or objects in it. A large number of simple features are defined in the literature, but they generally fall into basic categories (shape, size, brightness, and texture).

Image feature extraction is about summarizing/ converting raw image data into expressive representations that are more informative or show better association with an underlying phenomenon. The objective of such conversion is to highlight or make explicit in the data their most relevant elements with regard to a given task.
Also, we generally seek invariance properties so that the extraction result does not vary according to chosen (or specified) conditions. This implies finding objects regardless of their position, orientation or size. so that feature extraction techniques should find shapes reliably and robustly whatever the value of any parameter that can control the appearance of a shape.

Image features are used to compare two images or find similar landmarks or shared objects between multiple images. They can be either global features that describe the image as whole or local features that describe a part of any size of the image.

- **Global Features:**
  A very basic global image feature is the image histogram, which illustrates the probability distribution of the pixel/voxel values in the image, for each possible value, the number of occurrences is counted in the image. Global features, such as the shape of the histogram, can be used, for instance, to distinguish between classes of images, e.g., hand and skull radiography (see figure 1.14).

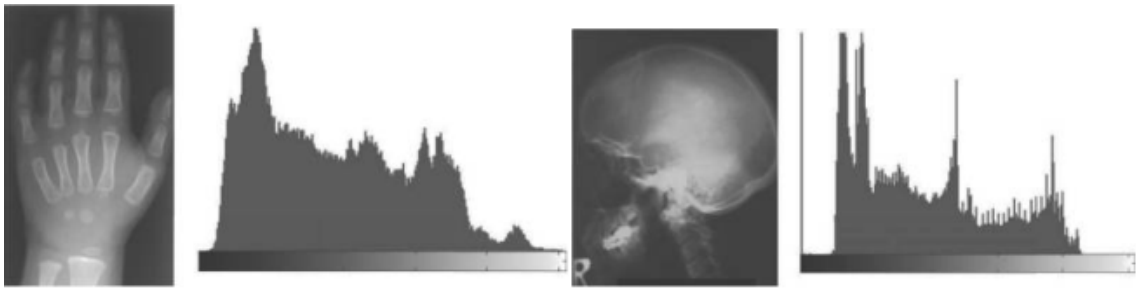


Figure 1.14: Images and their histograms. [32]

- **Local Features:**
  Local features describe only a part of the image at a certain spatial position. Most are created in two separate steps:

  I **features detection:** in which points of interest (POIs) are localized.

  II **features description:** for each of the detected points, a description of this position (possibly including some surrounding areas) is created.

Since images can be acquired under different conditions like scale and rotation, certain invariance against these changes is needed for both detector and descriptor.

Whether they are global or local, feature extraction methods can be classified based on their level of abstraction (or difficulty) into one of three broad groups:

I Classical, hand-crafted (also manual) features descriptors

II Advanced, latent-features representations

III Deep (also automated) features end-to-end learning methods (it's the most recent approach and our main interest area)

4.3.1 Classical, hand-crafted features descriptors

In general, these may refer to rudimentary (primitive or basic) features such as *image gradients* as well as fairly sophisticated features from elaborate algorithms such as the *histogram of oriented gradients feature (HOG)*.

More often than not, such features are designed by domain experts who have solid knowledge about the data properties and the demands of the task. Hence the name "*hand-crafted features*" is addressed.

Hand-engineering features for each task require a lot of manual labor and domain knowledge, and optimality is hardly guaranteed. However, it allows the integration of human knowledge of the real world and of that specific task into the feature design process, hence making it possible to obtain good results for the said task. These types of features are easy to interpret.
Note that it is not completely correct to call all classical features as being hand-crafted since some of them are general-purpose features with little task-specific tuning (such as outputs of *simple gradient filters*). [37]

Due to the chronological developments in feature engineering, some of these early feature extraction methods require more advanced and sophisticated algorithms than the others, thus they can be divided into two sub-groups "*low-level features detectors*" and "*high-level features detectors*".

I **Low-level features detectors:**
We shall define low-level features to be those basic features that can be extracted automatically from an image without any shape information (see figure 1.15).

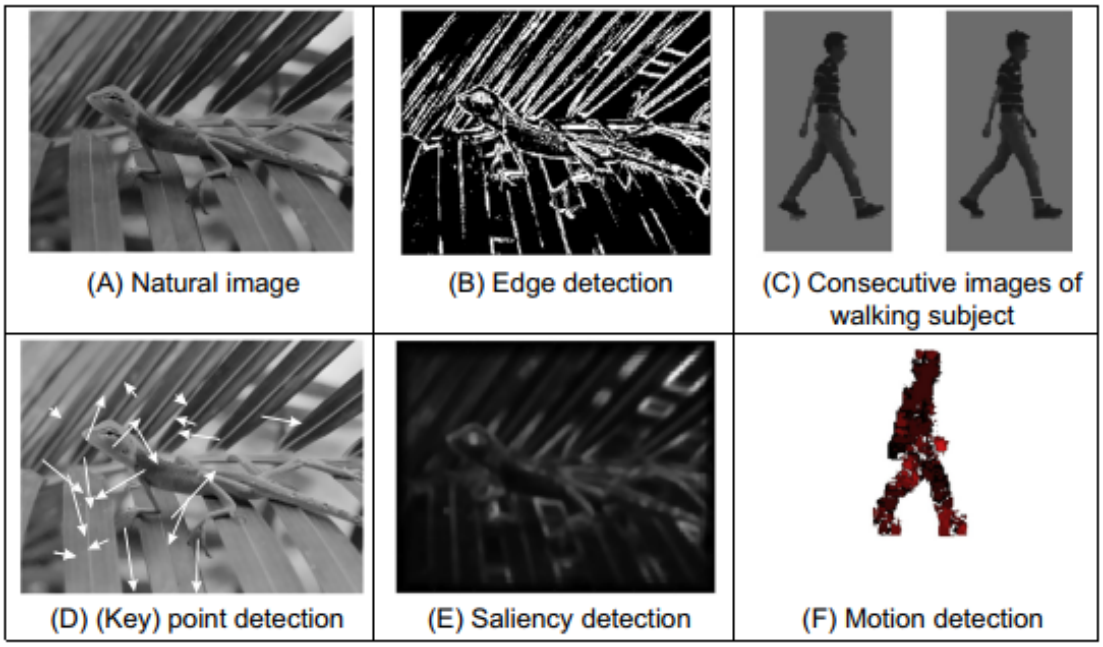


Figure 1.15: Low-level feature detection. [38]

- **Color features detection:** Color Coherence Vector (CCV), Histogram of Color Moments, Color Correlogram (CC).
- **Texture features detection:** Gray-level Co-occurrence Matrix (GLCM), Laws' Texture Masks, Gabor Filter, Steerable Gaussian Filter (SGF).

- **Edge detection:** highlights image contrast (difference in intensity), since detecting contrast can emphasis the boundaries of features within an image.
  - i First-order operators: Prewitt operator, Sobel operator
  - ii Second-order operators: Laplacian, MarreHildreth operator.
- **Phase congruency:** an alternative form of edge detection, operating on the considerations of phase (a.k.a. time).
- **Localized feature extractors:** two main areas are covered here, a traditional one that aim to estimate curvatures and corners, and a more modern area that employs region or patch-based analysis:
  - i **Detecting image curvature (corner extraction):** Moravec and Harris detector.
  - ii **Feature point detection; region/patch analysis:** Scale invariant feature transform (SIFT), Speeded up robust features (SURF), The Features from Accelerated Segment Test (FAST).

II **high-level features detectors:** concerns finding shapes and objects in images, whether they are fixed in shape (such as a segment of bone in a medical image) or shapes that can deform (like the shape of a walking person).

- **High-level feature extraction: fixed shape matching**
  - i **Thresholding and subtraction:** both are techniques to separate the objects from the background.
  - ii **Template matching:** a simple process, in which we try to match a template to an image, where the template is a sub-image that contains the shape we are trying to find.
  - iii **Feature extraction by low-level features:** combine a variety of low-level features like SIFT, SURF, BRIEF and HOG.
  - iv **Hough transform (HT):** introduced by hough 1962, it has been used in particular to extract lines, circles and ellipses (or conic sections).
- **High-level feature extraction: deformable shape analysis**
  - i **Deformable shape analysis:** including "*deformable templates*" and more advanced approach "*parts-based object analysis*".
  - ii **Active contours (or snakes):** a completely different approach where an active contour (a set of points) aims to enclose a target feature.
  - iii **Shape Skeletonisation:** a skeleton is a central axis to a shape determined by a distance transform.

4.3.2 Advanced, latent-features representations

Sometimes the feature space has some underlying properties that can not be extracted with hand-crafted features especially when the dataset is of high dimensions, therefore, it is desirable to discover latent structures through some automated process. Since these structures exist but are hidden in a new (sub)-space, they are called latent features (from the Latin word "latere").

Latent representations may expose the unobserved properties of data that exist but cannot be directly measured from the original data. These features usually seek a specific structure such as sparsity, decorrelation of reduced dimension, low rank, etc. Here are some of the most important latent feature types and the underlying extraction processes:

- **Principal Component Analysis (PCA):**
  PCA attempts to find a low-dimensional linear subspace that accounts for most of the variation in the data. It employs orthogonal transformations to convert a set of correlated variables into a set of linearly uncorrelated variables.

- **Kernel Principal Component Analysis (KPCA):**
  Kernel PCA tries to find a low-dimensional non-linear subspace if it existed, which is a powerful technique that outperforms the standard PCA.

- **Multidimensional Scaling (MDS):**
  MDS allows us to visualize similarities between samples in a dataset. Thus MDS is also a type of non-linear dimensionality reduction that tries to preserve distances in a reduced dimension space. There are four types of MDS algorithms: classical, metric, non-metric, and generalized MDS.

- **Isomap:**
  Isomap extends MDS in the sense that it uses geodesic distances in place of straight-line Euclidean distances between samples, such property makes it one of the most widely used non-linear dimensionality reduction methods.

- **Laplacian Eigenmaps:**
  A computationally efficient approach to non-linear dimensionality reduction that uses information from the neighboring data points, that are part of a weighted graph with a set of edges connecting the neighboring data points. It can be outlined in three steps: graph construction, edge Weighting, and computing the optimal embedding.

4.3.3 Deep features end-to-end learning methods

Introduction: Early motivations and challenges In the last two decades, CV researchers have focused on manually defined pipelines for extracting good image features, so image feature extractors such as SIFT and HOG were the standard. Although, those manually designed features require a great deal of human time and effort to handle a complex task.

A major difficulty that arise in such complex real-world applications is that the factors of variation that explain the observed data influence every single piece of it; the word "*factors*" simply to refer to separate sources of influence. For example when analysing an image of a car, the factors of variation include the position of the car, its color, and the angle and brightness of the sun, so each one of those can influence the other and confuse the system.
Thus most applications require us to disentangle those factors and discard the ones that we do not need. Eventually, it becomes a difficult challenge to extract such high-level, abstract features from raw data.

Deep learning solves this central problem in representation learning by extracting abstract representations that are expressed in terms of other simpler representations. Figure 1.16 shows how a DL system can represent the concept of an image of a person (such as corners and contours) by combining simpler concepts (such as edges).

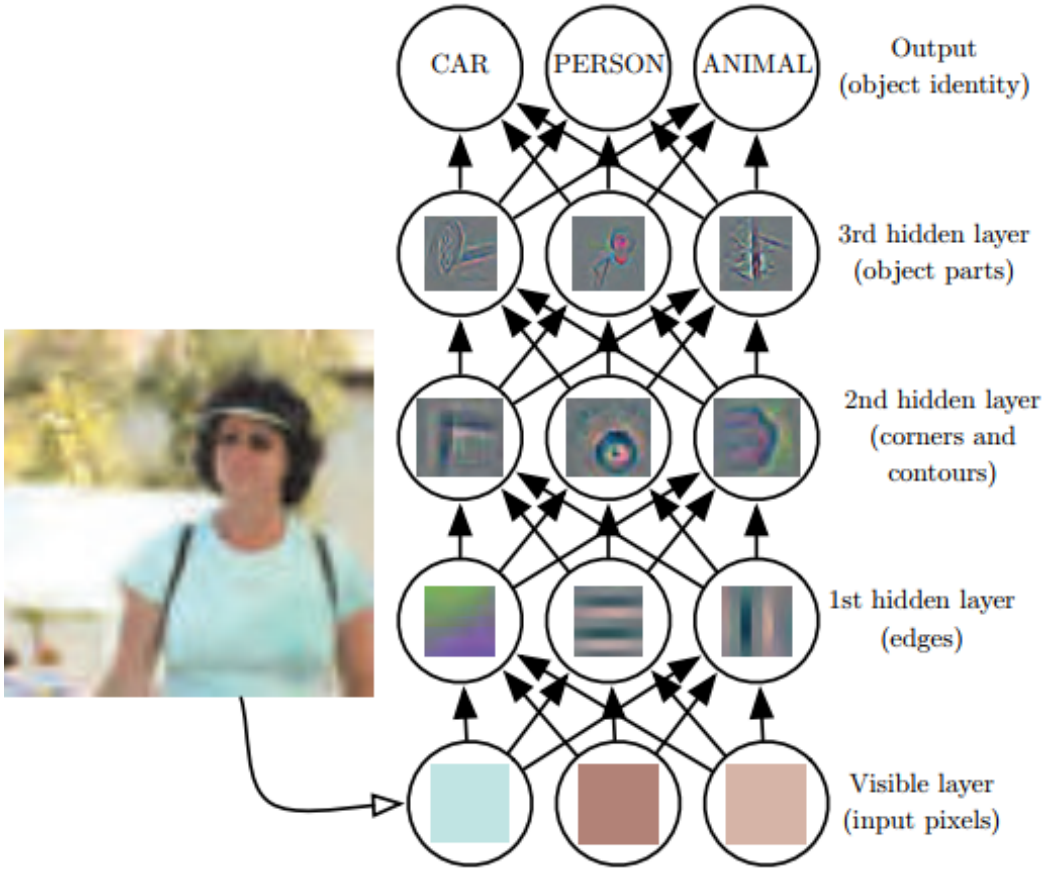


Figure 1.16: Illustration of a deep learning model in which the extracted features in each layer are visualised. Early layers contains simple features (e.g. edges, corners), the deeper the layer is the more abstract the feature gets, means complex representations are a combination of the simple ones. [1]

The recent developments in deep learning research have extended the reach of traditional ML models by incorporating automatic feature extraction in the base layers. The models that automatically learn and extract features are manu-

ally designed in terms of the architecture, so the manual work is still there, just abstracted further into the belly of the modeling beast.

History: Where did deep features come from! SIFT and HOG went a long way toward extracting good image features. However, the real achievements in computer vision have come from a completely different direction "*deep neural network models*".
The breakthrough had place at the ILSVRC (ImageNet Large Scale Visual Recognition Competition) 2012, where a group of researchers from the University of Toronto with the lead of *Alex Krizhevsky* has presented their 13 layers deep CNN model dubbed "AlexNet" (A. Krizhevsky et al., 2012 [17]) and nearly halved the error rate of the previous year's winner.
In the later organized ILSVRC competitions, even deeper models have been introduced and proved that the deeper it goes the better the performance it gets as the case of the winner of ILSVRC 2014 "GoogLeNet" that has 22 layers (C. Szegedy et al., 2014 [20]).

Method: How do we get deep features! Deep representations are obtained by passing raw input data with minimal preprocessing through a learned neural network, often consisting of a stack of convolutional and/or fully connected layers. As the input is propagated through each network layer, different data representations are obtained at abstract higher-level concepts. These networks are being trained iteratively by minimizing a task-specific loss that alters the parameters/weights in all layers.
For example, the specially designed convolutional layers of CNN allow it to extract translation-invariant features from images while the max-pooling layers of CNN help to reduce the parameters to be learned. [37]

A learned neural network is a deep neural network (two or more layers) that can learn deep features on its own. For that specific task, "*Convolutional Neural Networks (CNNs)*" and "*Recurrent Neural Networks (RNNs)*" are the most successful, although CNNs are designed for spacial images and RNNs for sequential data.

On the surface, the mechanism of stacked neural networks appears very different from the image gradient histograms of SIFT and HOG. But visualization of AlexNet shows that the first few layers are essentially computing edge gradients and other simple patterns, much like SIFT and HOG. Subsequent layers combine local patterns into more global patterns. The result is a feature extractor that is much more powerful than what came before. [39]

Deep learning systems can be thought of as multiple stages of applying linear operators and piping them through a non-linear activation function. [40]

Computational resources: What made it possible now! Until the onset of this decade, these methods were severely handicapped by a dearth of large-scale data and large-scale parallel computing hardware to be leveraged sufficiently.

We now have access to datasets that are large enough and graphics processing units (GPUs) that are capable of large-scale parallel computations. This has allowed an explosion of neural image features and their usage [41]. [37]

Advantage: Deep features' greatest use! Representation learning is particularly interesting because it provides one way to perform unsupervised and semi-supervised learning. We often have very large amounts of unlabeled training data and relatively little labeled training data.
Training with supervised learning techniques on the labeled subset often results in severe over-fitting. Semi-supervised learning offers the chance to resolve this over-fitting problem by also learning from the unlabeled data. Specifically, we can learn good representations for the unlabeled data, and then use these representations to solve the supervised learning task. [1]

Computer vision has now pivoted toward learning features from data, where new feature representations are typically learned using CNNs. The learned features in layers closer to the input layer can function as general feature extractors and be task-specific in layers closer to the task layer. [37]
Leveraging this observation, several off-the-shelf networks have been created and found extremely effective in many visual computing tasks, leading to tremendous gain in performance. The outstanding records in ILSVRC's multi-task challenges back up this claim. see (Krizhevsky et al., 2012 [17]; Zeiler & Fergus, 2013 [18]; Sermanet et al., 2014 [42]; Simonyan & Zisserman, 2014 [21]; C Szegedy et al., 2015 [20]) for further knowledge.

Transfer Feature Learning of CNNs: In the previous sections, we have fairly explained both the *CNNs* and *transfer learning* concepts. Now as we have introduced *deep features extraction*, we will bring all those elements together to explain *the application of transfer learning on CNNs*, as this is the core of the practical part of this work.

In practice, training an entire Convolutional Network from scratch (with random initialization) is rare as:

I It is very time-consuming and requires many computation resources

II It is relatively rare to have a dataset of sufficient size to train a ConvNet

Therefore, instead, it is common to pre-train a ConvNet on a very large dataset (e.g., ImageNet, which contains 1.2 million images with 1000 categories), and then use the ConvNet either as an initialization or a fixed feature extractor for the task of interest [43]. There are mainly two major transfer Learning scenarios, which are listed as follows: [37]

I **ConvNet as a fixed feature extractor:** In this scenario, we take a ConvNet pre-trained on ImageNet, remove the last fully connected layer, then treat the rest of the ConvNet as a fixed feature extractor for the new dataset. With the extracted features, we can train a linear classifier such as Linear SVM or

logistic regression for the new dataset.
This is usually used when the new dataset is small and similar to the original dataset. For such datasets, training or fine-tuning a ConvNet is not practical as ConvNets are prone to over-fitting to small datasets. Since the new dataset is similar to the original dataset, we can expect higher-level features in the ConvNet to be relevant to this dataset as well.

II **Fine-tuning the ConvNet:** The second way is to not only replace and retrain the classifier on top of the ConvNet on the new dataset but also fine-tune the weights of the pre-trained network using back-propagation.
The essential idea of fine-tuning is that the earlier features of a ConvNet contain more generic features (e.g., edge detectors or color blob detectors) that should be useful in many tasks, but later layers of the ConvNet become progressively more specific to the details of the classes contained in the original dataset.
If the new dataset is large enough, we can fine-tune all the layers of the ConvNet. If the new dataset is small but different from the original dataset, then we can keep some of the earlier layers fixed (due to overfitting concerns) and only fine-tune some higher-level portions of the network.

5 Conclusion

In this chapter, we presented brief but concise definitions from **deep learning** and its fundamental concepts to **computer vision** and its most recent methods, we have tried not only to mention these foundations separately but to build a bridge of knowledge between them to understand the goal of each one and its position in the chronological development of this science.

We have passed so far by the neural networks which are the stepping stone of deep learning, CNNs which is a more advanced concept capable of treating huge data, which eventually led to the emergence of transfer learning that helped a lot with the large access and applications of deep learning.

We then jumped to computer vision and image analysis, a different field that becomes reliable on DL and its approaches. All that to come to understand the process and the importance of deep feature extraction in image classification, a promising application that can bring lots of improvements to many life domains.

2
State of the art

In this chapter, we will select and review some notable approaches that discuss similar aspects to those of our goal thesis. As our thesis is a more cancer-oriented analysis approach, we will consider and pick some state-of-the-art studies on different cancer types (Brain cancer, breast cancer, lung cancer, and skin cancer) and discuss each group separately to better analyze their chronological development and the major contributions that have been brought out to each.

As there is an emerging applicability into the cancer analysis, we try to mention only the major ones that are heavily studied and contrasted in their methods in order to get an overview of as much methods as possible.

The selected papers will be stated in a summary Table 2.1 to get a summary and a comparative global view by mentioning all major points and specifications of the approaches.

1 Deep features related works on Brain Tumor detection

Amin et al. (2019) [44] methodology is based on fusing score vectors from Alex and Google CNN networks and the final vector is to be fed into multiple classifiers (KNN, SVM, LG, etc.) to detect brain tumors, both MRI (BRATS[1] datasets from 2013-2017) and CT (ISLES[2] 2018 dataset) images have been used for training and validation, as a first step image data are normalized and segmented. The presented method evaluated individual score vectors, as well as the fused vector which outperformed as 0.96 ACC (fused vector) compared to 0.92 ACC (single

[1]More about BRATS Datasets (2012-2018): https://www.med.upenn.edu/sbia/miccai-brats-2018-previous-brats-challenges.html

[2]More about ISLES 2018 challenge: http://www.isles-challenge.org

vectors) on the BRATS dataset and KNN, performed better than other classifiers in general.

Basheera et al. (2019) [45] approach was more simple but still efficient, they've used slow CNN (CNN-S) architecture on segmented MRI images from the AANLIB[3] Dataset by Harvard university (which contains only 66 MRIs), the CNN-S extracted features and classify the images in an embedded setting with the two-stage fully connected layers. The 8-layers deep model achieved 0.90 and 1 ACC on two small derived datasets from the original one.

1.1 Comparison

Therefore, for brain cancer analysis base-CNN methods, we've reviewed two approaches of the same year, Amin's [44] was more sophisticated with more steps and experiments with two deep CNN networks which gave it an advantage over S. Basheera's [45] which was more straight and direct with the limitation of small data collection. Actually, to prove that with a couple of improvised steps and the right tools we can always enhance the performance of the basic idea.

2 Deep features related works on Breast cancer recognition

Jiao et al. (2016) [46] proposed a 15-layers depth CNN to extract both middle and deep features to classify breast mammograms on the DDSM[4] dataset. The DDSM images were first resized (227x227), ROIs (region-of-interest) were extracted and normalized then fed into the CNN network, and middle and deep features were extracted from conv5 and fc7 respectively and fed each in an SVM classifier in a hybrid setting as 2-steps correct judgments of the decision process. Competitive classification accuracy of 96.7% has been achieved.

Nahid & Kong (2017) [47] carried out a histopathological image classification on "BreaKHis[5] dataset" with a 9-layers CNN, the approach is based on utilizing hand-crafted features "Histogram information (Hist.)" and "Local Binary Pattern (LBP)" along with the raw data image as an input to the proposed CNN. Better overall performance is achieved when using the raw image data along with (Hist.) information than with (LBP) information, a state-of-the-art accuracy on the "BreaKHis dataset" was achieved as 96%.

Spanhol et al. (2017) [48] worked on the same previous dataset (BreaKHis) with the AlexNet CNN of 8-layers depth, They've extracted features from 3 different layers (fc6, fc7, and fc8) and carried out different classification experiments with

[3]More about AANLIB: http://www.med.harvard.edu/AANLIB/

[4]More about DDSM: https://wiki.cancerimagingarchive.net/display/Public/CBIS-DDSM

[5] More about BreaKHis dataset: https://web.inf.ufpr.br/vri/databases/breast-cancer-histopathological-database-breakhis/

Logistic Regression to observe the impact of combining different deep features. Results presented that fc6 and fc7 features performed individually better than fc8's and a combination of two of them even slightly improved the accuracy at different patch levels. Results were overall around 85%.

Cao et al. (2019) [49] conducted a comparative study of several state-of-the-art CNN models to systematically evaluate their performance for both detection and classification of breast lesions, we will be only interested in their classification results. This study took 6 different Top-performing CNN architectures (AlexNet, ZFNet, VGG16, GoogLeNet, ResNet, and DenseNet) on a fairly-balanced collected dataset of "1043" ultrasound images manually annotated by experienced clinicians, all models were evaluated in 4 different scenarios (FULL-RI, FULL-FT, LROI-RI, and LROI-FT) in which data was either "full images or LROI (lesion regions of interest)" and CNN weights initialization were either by "RI (random initialization) or FT (features transfer)", in other words, the models were either trained from scratch or fine-tuned.

With this said, this extensive empirical evaluation has shown that (AlexNet, ZFNet, and VGG16) performed poorly on both "FULL-RI and LROI-RI" (trained from scratch) due to the curse-of-dimensionality problem that easily leads to over-fitting, in contrast to the other models (GooggleNet, ResNet, and DensNet) that performed better due to their different strategies to avoid over-fitting and DenseNet achieved the best accuracy of 80% on both LROI and full-sized images. With transfer learning from the large-scale annotated ImageNet[6], performance was further improved on all 4 scenarios and DenseNet again achieved the best results with 85% (on Full-sized images) and 87.5% (on LROIs). [Not included in the recapitulative table 2.1].

2.1 Comparison

Thus, for breast cancer based-CNN classification methods, we have seen three different approaches one of them was Z. Jiao's [46] on mammograms, and the two others on histopathological images (microscopic images) which is a more challenging task, regardless the data type z. Jiao's [46] and FA. Spanhol's [48] went in the same direction extracting and fusing deep features from different layers of the network with the only difference that they've used different classifiers, while AA. Nahid's [47] fused hand-crafted features with raw image data for CNN input which gave it the advantage to tackle the challenge, so considering the hand-crafted features along with CNN turned out to be a good choice.

Cao et al. [49] study broadens our vision about three critical points to consider when conducting breast CNN-based analysis: 1) on the input data which tends to give a more robust and efficient classification when utilizing LROIs, 2) on what architecture to use for breast lesion analysis which gave the superiority for the DenseNet and the ResNet, 3) on the advantage of transfer learning and fine-tuned models which were proven in terms of accuracy.

[6]More about ImageNet: http://www.image-net.org/challengesLSVRC/

3 Deep features related works on Lung Nodules segmentation

Xie et al. (2016) [50] approach proposed CT image classification for lung nodules by jointly using texture and shape features along with deep features from 9-layer DCNN based on the LeNet-5 model. The experiments conducted on the "LIDC-IDRI"[7] dataset, firstly they've to identify squared bounding boxes of lung nodules to obtain 9073 image patch from 1568 nodules, a resizing step of the patches into (32x32) is processed and then features are extracted from the DCNN's 7th layer (fc7), texture "GLCM" features and "Fourier descriptor" as shape features are extracted and combined then fed to a back propagation (BP) neural network for classification. Results have shown 87% accuracy which is considered competitive with the state-of-the-art methods.

Wang et al. (2017) [51] methodology for lung nodules classification is based on fusing deep features from the AlexNet CNN with hand-crafted features such as intensity and contrast features as well as the first and second-order filter features. The experiments conducted on the "JSRT[8] dataset" of chest radiography, firstly they've made some preprocessing steps including an enhancement for rib suppression, nodule region segmentation, and data augmentation to increase the performance of the CNN, deep features have been extracted from the FC layer of the AlexNet and fused with other hand-crafted features to be fed into a Random Forest classifier trained in 10-fold patch-based cross-validation, specificity of 96.2% was obtained.

Teramoto et al. (2017) [52] proposed a DCNN of 8-layers deep to classify lung cancer from cytological images, the experiments were conducted on an originally collected dataset of 298 cytological images, which were firstly cropped and resized to (256×256) and training images were augmented to avoid over-fitting, the image data were then fed to the DCNN and from its last layer, probabilities of cancer types (adenocarcinoma, squamous cell carcinoma, and small cell carcinoma) were obtained using a softmax function. The proposed DCNN achieved a 71.1% classification accuracy which is comparable to that of a cytotechnologist or pathologist.

Nibali et al. (2017) [53] proposed a framework based on the ResNet-18 model but with introducing a curriculum learning strategy along with transfer learning in a hybrid setting. The experiments were conducted on the LIDC/IDRI dataset, the image data were preprocessed including the normalization of Hounsfield units (HU), cropping and re-sampling of CT slices, and then fed to the ResNet-18, the model Was pre-trained on the CIFAR-10[9] dataset for better initialization of the weights, and a curriculum learning technique was followed for training on the new data which is increasing the difficulty of training by time by introducing the

[7] More about LIDC-IDRI dataset: https://wiki.cancerimagingarchive.net/display/Public/LIDC-IDRI

[8] More about JSRT dataset: http://db.jsrt.or.jp/eng.php

[9] More about CIFAR-10: https://www.cs.toronto.edu/~kriz/cifar.html

easy samples first which may improve parameter optimization. With test data being augmented, classification accuracy achieved was 89.9% which outperformed all similar state-of-the-art works.

3.1 Comparison

Hence, for lung cancer CNN-based classification methods, we have reviewed 4 approaches conducted on different data types from CT images to radiographic and cytological images, two of them were Xie's [50] and Wang's [51] that went to extracting and fusing deep with hand-crafted features and both obtained excellent results with an advantage to Wang's method probably for the carefully preprocessing of data. The other two methods were A. Teramoto's [52] & Nibali's [53] that proposed different architectures and classified with the softmax layer itself, both have obtained excellent accuracy with an advantage to the ResNet-based approach perhaps for its genuine framework that embedded three types of powerful learning techniques (residual, curriculum, and transfer learning) as well as the challenging of data of cytological data which don't allow a fair comparison here.
Therefore, the extraction/fusion approach is efficient for lung cancer classification and standalone deep CNNs can perform well also if they are carefully carried out and can even tackle more challenging data-related problems like A. Teramoto's [52].

4 Deep features related works on Skin Lesions detection

Yu et al. (2017) proposed a hybrid framework for skin lesion classification based on AlexNet and Fisher vector, the experiments are implemented on the ISBI[10] 2016 Melanoma Detection Challenge Dataset of 1279 annotated skin images, after preprocessing the data (data augmentation & normalization) sub-images of (227x227) are fed to the 8-layers deep AlexNet then features from FC6 and FC7 are extracted, then an FV (Fisher vector) encoding was performed to aggregate these features to build more invariant representations. Finally, the FV encoded representations are classified for diagnosis using a linear SVM, final results demonstrate that FC6 features with an "ACC=82.32%" perform better than FC7's, further improvements were observed "ACC=83.09%" when combining FV representations from multi-scale images.

Harangi et al. (2018) [54] presented an ensemble approach for skin lesions in which they aggregated 3 different architectures into one, they chose namely "AlexNet, VGG-16, and GoogLeNet" each of which is top-performing on the ImageNet contest. Their experiments were evaluated on the ISBI 2017 challenge dataset with a data augmentation step, the architecture considered removing the final fully-connected layer of these individual CNNs, then interconnecting them by inserting a joint fully connected layer followed by a softmax layer for the final pre-

[10]More about ISBI challenge: https://biomedicalimaging.org/2017/past-isbis/

diction. Comparative results have shown average accuracy of 84.8% for the proposed model with a clear advantage over individual and dual models.

Mahbod et al. (2019) proposed a hybrid approach for skin lesion classification employing multiple CNNs, 3 different models were considered namely "AlexNet, VGG-16, and ResNet-18", the setting of this framework introduced feature extraction from the last FC layers of each model (also last Conv layer for ResNet), then train multi-class non-linear SVM classifiers on these features, the classification scores were then combined and mapped to probabilities with Logistic regression (LR). The experiments were evaluated on the ISIC 2017[11] challenge dataset with minimal preprocessing steps (normalization, resizing, and augmentation), and outperforming results were achieved with 90.69% as average ACC.

4.1 Comparison

Thus, for skin lesion classification, we have seen three different studies on similar datasets, while Yu's [55] method was in a different direction than the others introducing FV encoding to optimize the features which achieved good results, Harangi's [54] frm[o]– Mahbod [56] went towards ensemble approaches introducing slightly different combinations of CNNs and completely different hybrid settings, they've both achieved better ACC with an advantage to Mahbod's [56] who combined the SVMs outputs instead of combining the architectures themselves.

5 General Synthesis

In this section, we group the previously reviewed papers altogether according to their domain application, in this table we highlight the major differences between them in several points (Type of data, pre-processing steps, method, architecture,..etc.), then we state the overall accuracy of each one in the last column to evaluate the performance of each method regarding the rest.

[11] More about ISIC challenge: https://challenge.isic-archive.com/landing/2017

Approach	Year	Data	Pre-processing	Method	Architecture	TL	Classifier	ACC
Brain Cancer								
Amin et al. [44]	2019	MRI/CT	Norm./Seg.	Multi-CNNs	Alex/GoogLe	Yes	Multi-classifiers	0.96 (BRATS 2013)
Basheera et al. [45]	2019	MRI	Seg.	CNN	CNN-S	Yes	Softmax	0.90/1
Breast Cancer								
Jiao et al. [46]	2016	MG	Res./Seg./Norm.	CNN (Mid+Deep levels)	15-layers CNN	Yes	Two-SVMs	0.96
Nahid & Kong [47]	2017	Hist.	No	CNN+Hist/LE	9-layers CNN	No	Softmax	0.96
Spanhol et al. [48]	2017	Hist.	No	CNN (3 Deep levels)	AlexNet	Yes	LG	0.85
Lung Cancer								
Xie et al. [50]	2016	CT	Seg./Res.	DCNN+GLCM	9-layers DCNN	No	BP-NN	0.87
Wang et al. [51]	2017	X-Ray	Enh./Seg./Aug.	CNN+Int.+Co	AlexNet	Yes	RF	0.69/0.96 (Sens/Spec)
Teramoto et al. [52]	2017	Cyto.	Crop./Res./Aug.	DCNN	8-layers DCNN	No	Softmax	0.71
Nibali et al. [53]	2017	CT	Norm./Crop./Res./	CNN+Cirr.	ResNet-18	Yes	Softmax	0.89
Skin Cancer								
Yu et al. [55]	2017	ELM	Aug./Norm.	CNN+FV	AlexNet	Yes	SVM	0.83
Harangi et al. [54]	2018	ELM	Aug.	Multi-CNNs	AlexNet/VGG-16/GoogLeNet	Yes	Softmax	0.84
Mahbod et al. [56]	2019	ELM	Norm./Res./Aug.	Multi-CNNs	AlexNet/VGG-16/ResNet-18	Yes	SVM/LR	0.90

Table 2.1: A summary table of the stated approaches and their full properties.

6 Conclusion

In this chapter, we have reviewed some of the most notable papers in different cancer detection areas (Brain Cancer, Breast Cancer, Lung Cancer, and Skin Cancer).
All of these are recent studies that explore different deep learning CNN networks with different settings, some rely on auto-generated deep image features while the others introduce hand-crafted features as a plus, some use softmax layers for classification and others use third-party classifiers like SVM, RF or BP-NN or even multiple-classifiers to produce more power.

Although the different domain-related challenges for every type of cancer, we can say that the most successful results were those that used fused features from different CNN networks and/or multiple classifiers.

In the next chapter, we will use the VGG-16 as a feature extractor with third-party classifiers (SVM, MLP, and RF) to explore this approach to Breast Cancer classification.

3 Implementation

1 Introduction:

In this chapter, we will carry out a process of deep feature extraction for medical image classification as an application example to experimentally justify the efficiency of this novel approach.
As in this work, we chose to work on INbreast [57] dataset for breast cancer classification, for this mission we needed the right CNN model that can bring out the best possible performance.

It is noteworthy that convolution neural networks have done well so far and credits partly go to the ImageNet[1] which is a dataset of over 14 million images belonging to 1000 classes, because of this huge large-scale image dataset, DL engineers and researchers have been since then implementing pre-trained CNN models on ImageNet and fine-tuning them on their datasets in all application domains, that's what we've talked about earlier in the first chapter as "Transfer learning" because building CNN from scratch is not an option with poor data at hand.
Another challenging aspect of TL is that even with the fine-tuned weights we potentially can have bad features among those we extract which affect the overall accuracy, which requires some techniques on several levels from "data preprocessing" such as data augmentation (by image flipping and rotation...etc.) to gain more training knowledge, to "architecture design" such as adding new layers on top of the model (FC layers, max-pooling layers, heatmaps..etc.).

In our experiments, to ease these time-consuming struggles and for our poor data and resources, we build our classification process on top of deep features extracted from a VGG-16 model pre-trained on ImageNet and fine-tuned to clas-

[1] More about ImageNet: http://www.image-net.org/challengesLSVRC/

sification on CBIS-DDSM[2] then with transfer learning to classify INbreast.

1.1 Starting point: end-to-end Li Shen's study

Our experiments are based on one of the CNN models used in a study by L. Shen et al. (2017) [58], the prestigious team from the USA carried out this study as their entry to the DREAM2016[3] Digital Mammography challenge, they've then improved their method and presented it as their NIPS17[4] (Neural Information Processing Systems) conference workshop.

Briefly, they have used different pre-trained CNN models (VGG-16, VGG-19, ResNet-50, and YaroslavNet) as patch-based classifiers which classify mammograms based on specific regions or slides which is a common strategy to address when having small lesions and complex data, then they have converted those patch-based to whole image classifiers by adding some layers on the top (heatmap, residual blocks..etc.). (See figure 3.1)

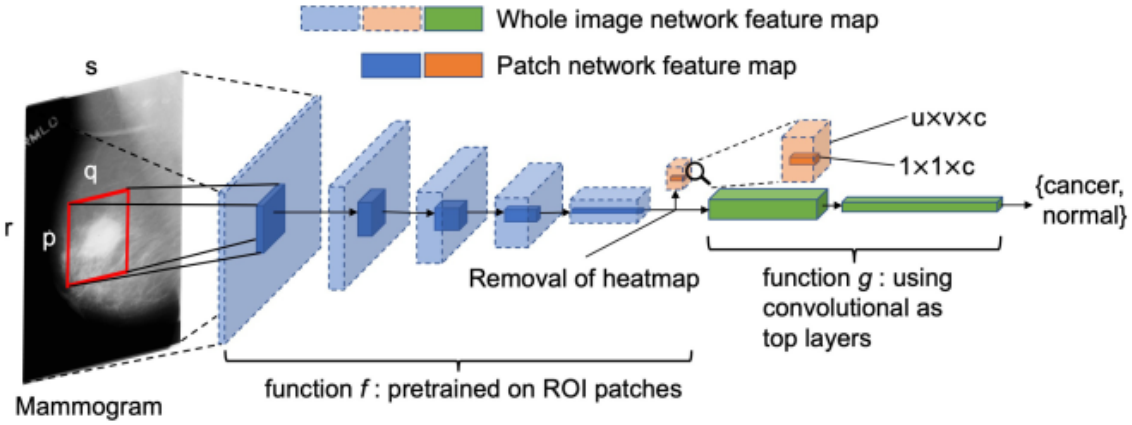


Figure 3.1: Converting a patch classifier to an end-to-end trainable whole image classifier using an all convolutional design where they considered removing the heatmap to improve information flow and convolutional layers as top layers; the magnifying glass shows an enlarged version of the heatmap. [58]

They have conducted their study on a recent version of the "Digital Database for Screening Mammography" (DDSM) called CBIS-DDSM dataset, after getting excellent results comparable to state-of-the-art they have fine-tuned some of their models on the INbreast dataset to test the transferability of the whole image classifiers and it worked well. Consider checking their GitHub repository for further details [59].
Note that the INbreast dataset [57] is a more recent public database for mammograms that contains FFDM images as opposed to digitized film images. These images have different intensity profiles from the DDSM images, which can be

[2] More about CBIS-DDSM: https://wiki.cancerimagingarchive.net/display/Public/CBIS-DDSM
[3] More about DREAM challenges: http://dreamchallenges.org
[4] More about NIPS17: https://nips.cc/Conferences/2017

visually confirmed by looking at two example images from the two databases (see figure 3.2). Therefore, INbreast provides an excellent opportunity to test the transferability of a whole image classifier onto an independent dataset.

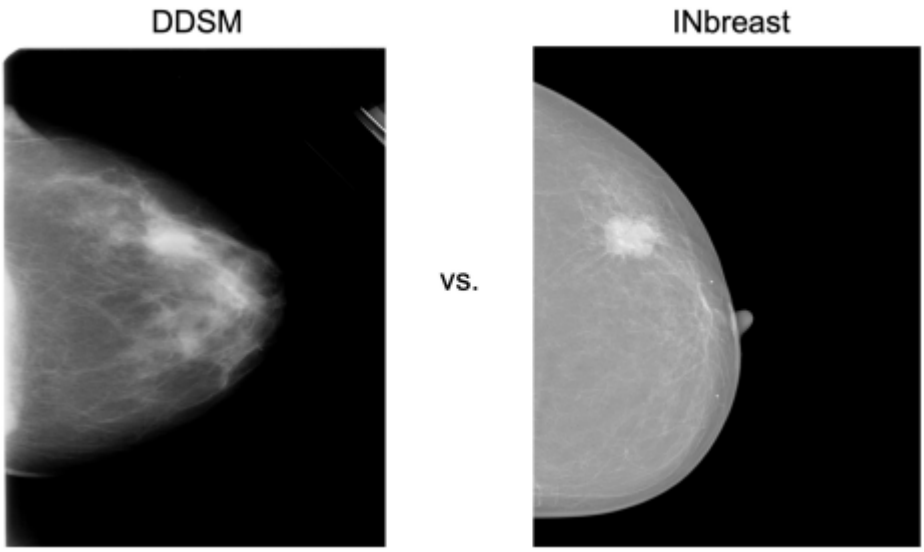


Figure 3.2: Comparison of representative mammograms from DDSM and INbreast. [58]

So we have considered using their VGG-16 to extract features from our INbreast dataset for the following reasons:

I The model is pre-trained on ImageNet which gives the model a strong ability to learn and generalize from other data

II The model is trained on DDSM mammograms so it has domain-specific knowledge as we will carry in our experiments for the same kind of data

III It was additionally fine-tuned on INbreast so the weights are updated and optimized

Hence the model has learned strong representations from a large image set including breast mammograms, so the odds are in favor to extract useful features without the earlier-mentioned TL-related challenges.

1.2 INbreast dataset

The INbreast dataset is a mammographic dataset, with images acquired at a Breast Centre, located in a University Hospital (Centro Hospitalar de São João (CHSJ) in Porto, Portugal). It is a recent public database for mammograms that contains full-field digital mammography (FFDM) images.
INbreast has a total of 115 cases were collected, from which 90 have two images (MLO and CC) of each breast and the remaining 25 cases are from women who had a mastectomy, and two views of only one breast were included. This sums to a total of 410 images. Eight of the 91 cases with 2 images per breast also have

images acquired in different timings (follow-up). There are a total of 116 masses among 107 images (≈ 1.1 masses per image), the overall distribution of **benign/-malignant** cases is shown in figure 3.4(b).

INbreast recording includes two breast views: (See figure 3.3)

- The CC (craniocaudal) view: a top to bottom view
- The MLO (mediolateral oblique) view: a side view

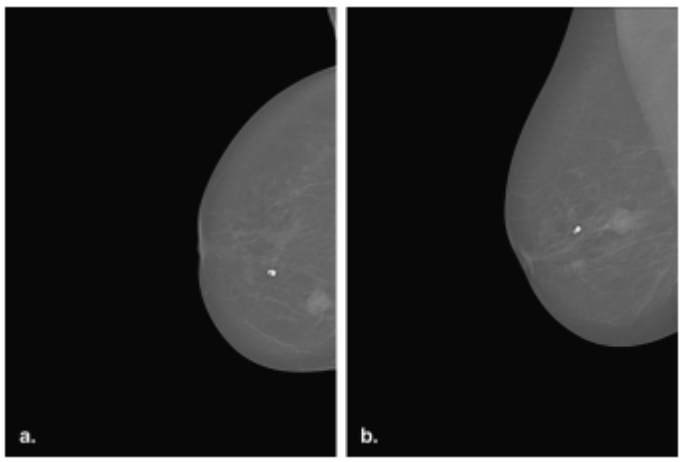


Figure 3.3: Database examples: multiple findings. (a) Craniocaudal view of the right breast; (b) mediolateral oblique view of the right breast. [57].

To standardize the terminology of the mammographic report, the assessment of findings, and the recommendation of action to be taken, the American College of Radiology (ACR) has developed the Breast Imaging Reporting and Data System (BI-RADS) scale [60].

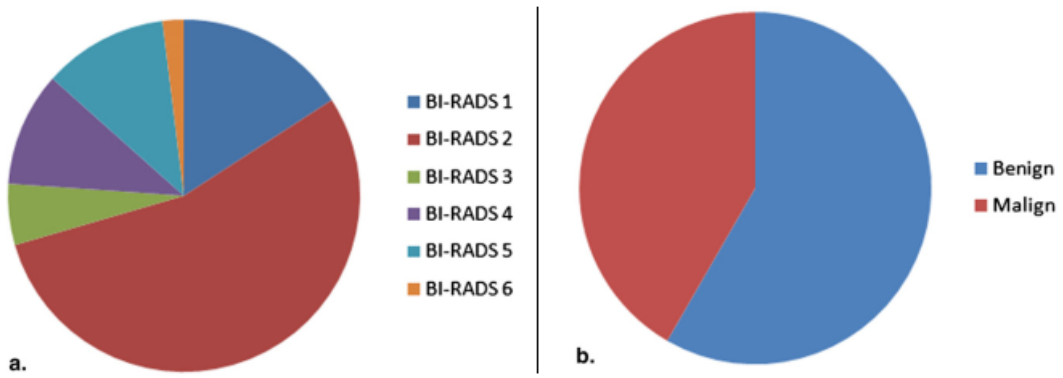


Figure 3.4: Charts of (a) the BI-RADS image distribution (b) benign/malignant cases distribution. [57].

Based on level of suspicion, lesions can be placed into one of six BI-RADS categories: (See figure 3.4(a))

- category **0**: exam is not conclusive;
- category **1**: no findings;
- category **2**: benign findings;
- category **3**: probably benign findings;
- category **4**: suspicious findings;
- category **5**: a high probability of malignancy;
- category **6**: proved cancer.

Several types of lesions (masses, calcifications, asymmetries, and distortions) are included (See figure 3.5. Accurate contours made by specialists are also provided in XML format.

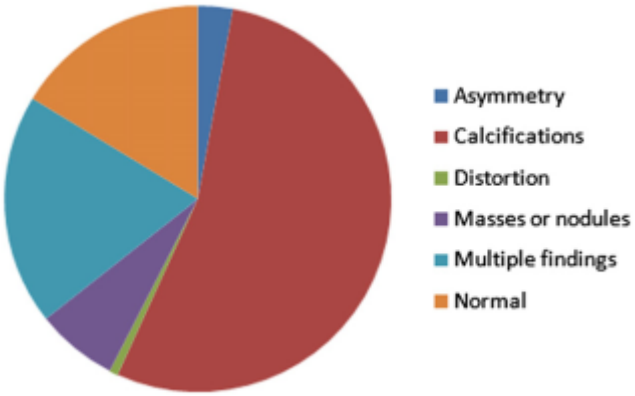


Figure 3.5: Chart describing the findings in the INbreast database. [57].

1.3 The VGG-16 model

VGG-16 is a convolutional neural network model proposed by K. Simonyan and A. Zisserman from the University of Oxford in this paper [21]. The model achieved 92.7% top-5 test accuracy in ImageNet. It was one of the famous models submitted to ILSVRC-2014[5]. It improves AlexNet by replacing large kernel-sized filters (11 and 5 in the first and second convolutional layer, respectively) with multiple 3×3 kernel-sized filters one after another. VGG16 was trained for weeks and was using NVIDIA Titan Black GPUs. VGG-16 architecture is detailed in figure 3.6.

[5]More about ILSVRC-14: http://www.image-net.org/challenges/LSVRC/2014/results

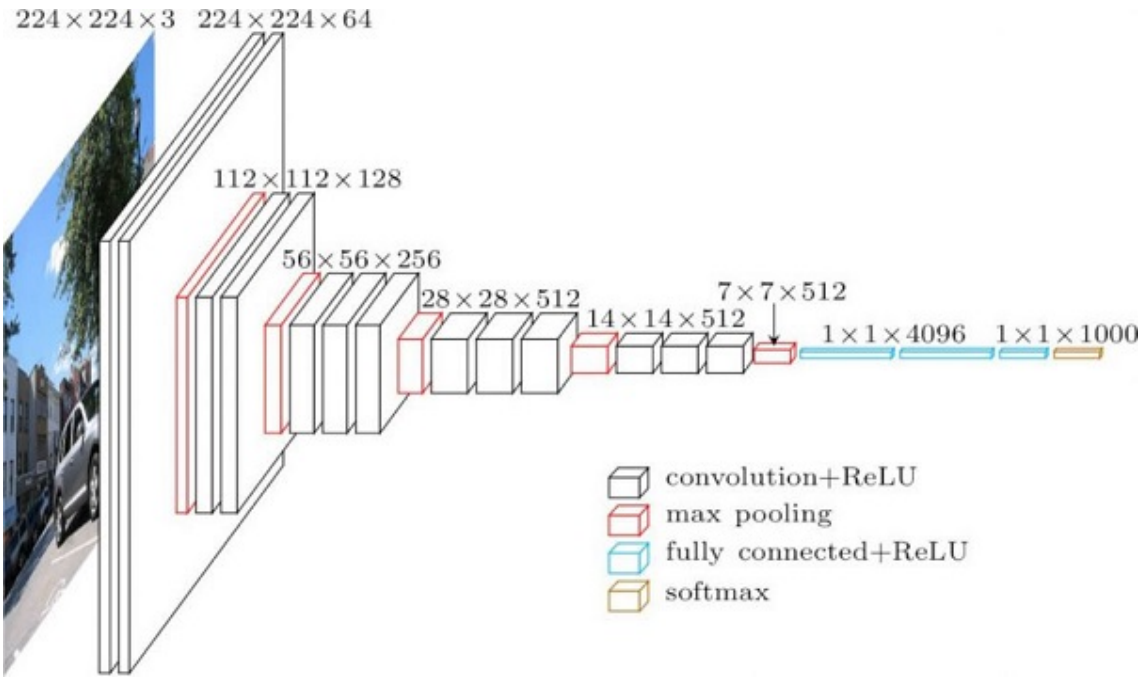


Figure 3.6: VGG-16 architecture diagram.

1.4 Development environment: Google Colaboratory

Google Colaboratory[6] (or Colab) is a Google research project created to help disseminate ML education and research, it is based on a Jupyter notebook environment that requires no setup to use and runs entirely in the cloud.
Colab grants users access to a wide range of GPU (graphics processing unit) like the "NVIDIA Tesla K80" GPU and even a TPU (Tensor processing unit) freely for 12 hours per runtime, this should be enough for most small and medium deep learning projects, Colab also allows users to access, edit, download and save files from and into Google Drive which serves greatly as Colab delete local data every 12 hours when the free runtime is over, so local data (variables, files,.etc.) get erased and a new 12h runtime gets initialized.

[6]Link to Colab: https://colab.research.google.com/

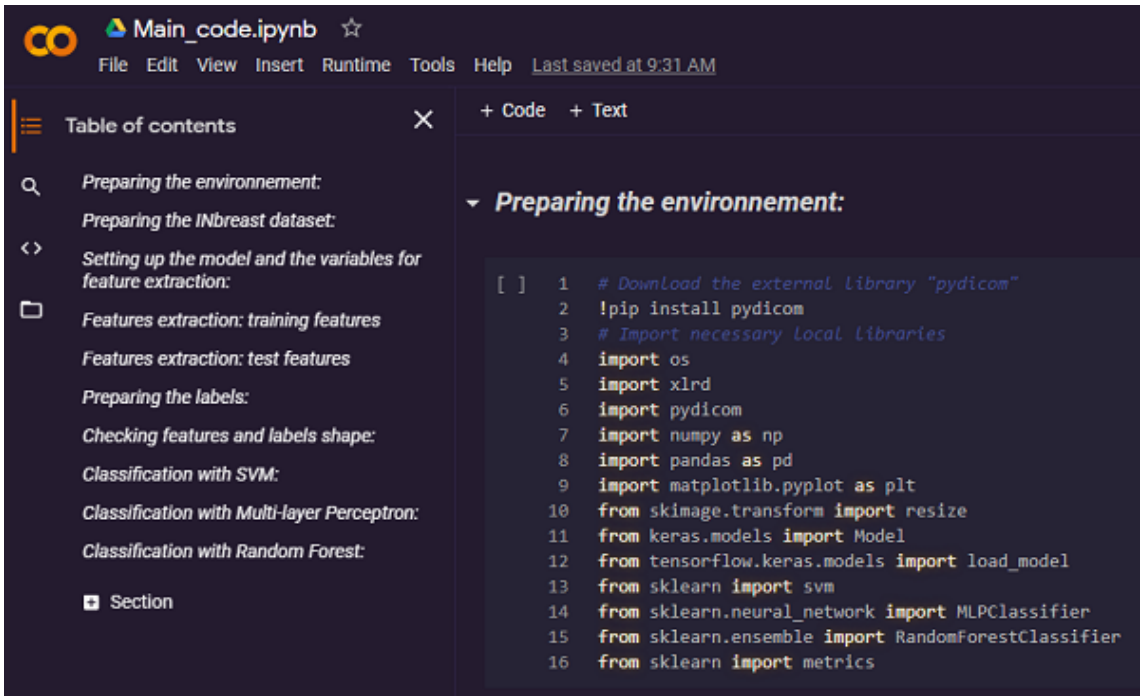


Figure 3.7: Screenshot of Colab environment with dark theme.

2 Code implementation

2.1 Preparing the environment

```
# Download the external library "pydicom"
!pip install pydicom
# Import necessary local libraries
import os
import xlrd
import pydicom
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from skimage.transform import resize
from keras.models import Model
from tensorflow.keras.models import load_model
from sklearn import svm
from sklearn.neural_network import MLPClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn import metrics
```

Listing 1: Import some libraries

In this code 1, we install the external "pydicom" library that we will need to

read (.dcm) images, we import other local libraries to the working directory for later use.

2.2 Preparing the INbreast dataset:

In this code 2 we verify the integrity of our data, we have made it into two sets "Training" and "Test", on which we followed the aforementioned L. Shen's study in their data partitioning to avoid potential performance decrease.
They've stated in their paper that "Bi-Rads=3" assessment is not typically given at screening, therefore they ignored "23 images" and so we did, it remains "387 images" with other Bi-Rads values (1, 2, 4a, 4b, 4c, 5, 6) from which we addressed "280 images" for training and "107 images" for the test, the output of code confirms same set sizes, The code for this part and its output is the following: (See figure 3.8).

```
# List of "training" images
tr_data = os.listdir('/content/drive/MyDrive/BRIKI PFE 2020/Datasets/INbreast_for_end-to-end/Training_Data')
# List of "test" images
ts_data = os.listdir('/content/drive/MyDrive/BRIKI PFE 2020/Datasets/INbreast_for_end-to-end/Test_Data')
# Nnumber of "training" images
print ('Training data:',len(tr_data),'images')
# Nnumber of "test" images
print('Test data:',len(ts_data),'images')
```

Listing 2: Training and test image data

```
Training data: 280 images
Test data: 107 images
```

Figure 3.8: Output of code 2: Training and test image data.

2.3 Setting up the model and the variables for feature extraction

In this part 3, we load the "`VGG-16`" by passing its path to the "`load_model`" function, then we display a summary to see each of its layers and how many trained parameters per layer and the more important the size of data fed to each one (See figure 3.9).
We chose the "`global_average_pooling2d_1`" layer as our feature extractor, from which we extract "1024" features for each image in the dataset.
Next, we initialize a 3D NumPy array of zeros to temporarily hold the image while before passing it to the network, another 1D NumPy array of zeros is used to concatenate all the features to be extracted.

We also set the path to both training and test data directories, and the output one to where we intend to save the features.

```
# Load the model
cnn_vgg = load_model('/content/drive/My Drive/BRIKI PFE 2020/Hypothesis/inbreast_vgg16_[512-512-1024]x2_hybrid.h5')
cnn_vgg.summary()
# Set a feature extractor from a specific layer
ft_ext_vgg = Model(inputs=cnn_vgg.input,outputs=cnn_vgg.get_layer('global_average_pooling2d_1').output)
# Set a 3D numpy array of zeros to temporarily stock an image
img_zeros=np.zeros(shape=(1152, 896,3))
# Set a 1D numpy array of zeros where to concatenate the extracted features
ft_zeros=np.zeros((1,1024))
# Set input paths
img_train='/content/drive/MyDrive/BRIKI PFE 2020/Datasets/INbreast_for_end-to-end/Training_Data'
img_test='/content/drive/MyDrive/BRIKI PFE 2020/Datasets/INbreast_for_end-to-end/Test_Data'
# Set the output path where to save extracted features and labels
output='/content/drive/MyDrive/BRIKI PFE 2020/Datasets/INbreast_for_end-to-end'
```

Listing 3: Loading the VGG-16 and variables

```
Model: "model_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
 ↪  Connected to
==================================================================
input_1 (InputLayer)         [(None, 1152, 896, 3 0
_________________________________________________________________
model_1 (Functional)         (None, None, None, 5 14714688
 ↪  input_1[0][0]
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 18, 14, 512)  262656
 ↪  model_1[0][0]
_________________________________________________________________
::      ::      ::      ::      ::      ::      ::      ::      ::
 ↪  ::      ::      ::      ::      ::      ::      ::      ::
::      ::      ::      ::      ::      ::      ::      ::      ::
 ↪  ::      ::      ::      ::      ::      ::      ::      ::
::      ::      ::      ::      ::      ::      ::      ::      ::
 ↪  ::      ::      ::      ::      ::      ::      ::      ::
_________________________________________________________________
dropout_13 (Dropout)         (None, 1024)              0
global_average_pooling2d_1[0][0]
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 2050
dropout_13[0][0]
==================================================================
Total params: 29,710,146
Trainable params: 29,689,666
Non-trainable params: 20,480
_________________________________________________________________
```

Figure 3.9: Output of code 3: Display of the architecture of the model

2.4 Features extraction: training features

```
for root, dirs, files in os.walk(img_train):
  # Browse images in a-z order
  for file in sorted(files):
    img1 = os.path.join(img_train, file)
    # Read an image and preprocess it
    img2 = pydicom.read_file(img1)
    img3 = img2.pixel_array
    img4=img3*(255/16383)
    img5 = resize(img4, [1152, 896], anti_aliasing=True)
    img_zeros[:,:,0]=img5
    img_zeros[:,:,1]=img5
    img_zeros[:,:,2]=img5
    img_final=np.expand_dims(img_zeros,0)
    # Here feature extraction
    ft_tr=ft_ext_vgg.predict(img_final)
    ft_zeros=np.concatenate((ft_zeros,ft_tr),axis=0)
ft_final=np.delete(ft_zeros, 0, 0)
# Save features array with the specified name
ft_path=os.path.join(output,'vgg_tr_ft')
np.save(file=ft_path,arr=ft_final)
```

Listing 4: Extraction of training features.

Next code 4 is to launch feature extraction from the training set with a for loop, we pre-process raw pixels before extracting features from by:

- **Normalization**: As INbreast images are coded on 14bits therefore we divide our pixels on a max value of "16383", and multiply by "255" to get our images coded on 8bits as our VGG is trained on the same encoding.
- **Resizing**: we then resize images to the same size as the VGG input layer [1152, 896].

Next is to add another dimension to the image to fit the input layer, we then pass it to extract features from and concatenate them in one array and save it on as "`vgg_tr_ft`".
Note that we have used "sorted(files)" to read the images in an alphabetic order to make sure they will match the right labels later, this is a simple trick yet deterministic.

2.5 Features extraction: test features

```
for root, dirs, files in os.walk(img_test):
  # Browse images in a-z order
  for file in sorted(files):
    img1 = os.path.join(img_test, file)
    # Read an image and preprocess it
    img2 = pydicom.read_file(img1)
    img3 = img2.pixel_array
    img4=img3*(255/16383)
    img5 = resize(img4, [1152, 896], anti_aliasing=True)
    img_zeros[:,:,0]=img5
    img_zeros[:,:,1]=img5
    img_zeros[:,:,2]=img5
    img_final=np.expand_dims(img_zeros,0)
    # Here feature extraction
    ft_tr=ft_ext_vgg.predict(img_final)
    ft_zeros=np.concatenate((ft_zeros,ft_tr),axis=0)
ft_final=np.delete(ft_zeros, 0, 0)
# Save features array with the specified name
ft_path=os.path.join(output,'vgg_ts_ft')
np.save(file=ft_path,arr=ft_final)
```

Listing 5: Test features extraction.

In 5, we repeat exactly the same process for test set images and save test features as a numpy array "`vgg_ts_ft`".

2.6 Preparing the labels

```
# Access and open labels "xlsx" file
location='/content/drive/MyDrive/BRIKI PFE 2020/Datasets/INbreast_for_end-to-end/Labels/labels_finalVersion.xlsx'
lb1 = xlrd.open_workbook(location)
lb2 = lb1.sheet_by_index(0)
lb3=lb2.col_values(0)
# Load labels in a numpy array
lbs_final=np.array(lb3)
# Split labels to training and test
tr_lbs=lbs_final[0:280] #280 labels
ts_lbs=lbs_final[280:387]   #107 labels

# Save labels arrays
tr_lbs_path=os.path.join(output,'tr_lbs')
ts_lbs_path=os.path.join(output,'ts_lbs')
np.save(file=tr_lbs_path,arr=tr_lbs)
np.save(file=ts_lbs_path,arr=ts_lbs)
```

Listing 6: Reading and splitting labels.

As we have our features ready, now in this code 6 we move to prepare labels by reading a (.xlsx) file of the full "387 images" labels prepared in the same order as the images to guarantee the right matching.
We pass the labels to a NumPy array and then split it in the same ratio as the images before, 280 for training and 107 for the test. We save the labels as "`tr_lbs`" and `ts_lbs`".

2.7 Checking features and labels shapes

```
# Load previously saved features and labels
vgg_tr_ft=np.load('/content/drive/MyDrive/BRIKI PFE 2020/Datasets/INbreast_for_end-to-end/vgg_tr_ft.npy')
vgg_ts_ft=np.load('/content/drive/MyDrive/BRIKI PFE 2020/Datasets/INbreast_for_end-to-end/vgg_ts_ft.npy')
tr_lbs=np.load('/content/drive/MyDrive/BRIKI PFE 2020/Datasets/INbreast_for_end-to-end/tr_lbs.npy')
ts_lbs=np.load('/content/drive/MyDrive/BRIKI PFE 2020/Datasets/INbreast_for_end-to-end/ts_lbs.npy')
# Print shapes
print('Size of training features:',vgg_tr_ft.shape)
print('Size of test features:',vgg_ts_ft.shape)
print('Size of training labels:',tr_lbs.shape)
print('Size of test labels:',ts_lbs.shape)
```

Listing 7: Features and labels shape.

```
Size of training features: (280, 1024)
Size of test features: (107, 1024)
Size of training labels: (280,)
Size of test labels: (107,)
```

Figure 3.10: Output of code 7: Features and labels shape.

Before we move on to classification, with this code 7 we first check the dimensions of the data and labels for both training and test sets to avoid any mismatch. We simply load the earlier saved arrays of features and labels, then we "print" each one's shape (See figure 3.10).

2.8 Classification with SVM

```
# Call SVM classif
svm_cls = svm.SVC()
# Training SVM
svm_cls.fit(vgg_tr_ft, tr_lbs)
# Testing SVM
svm_lbs_pred = svm_cls.predict(vgg_ts_ft)
# Print evaluation metrics
print("Accuracy:",metrics.accuracy_score(ts_lbs, svm_lbs_pred))
print("Precision:",metrics.precision_score(ts_lbs, svm_lbs_pred))
print("Recall:",metrics.recall_score(ts_lbs, svm_lbs_pred))
print("F1-score:",metrics.f1_score(ts_lbs, svm_lbs_pred))
metrics.plot_confusion_matrix(svm_cls, vgg_ts_ft, ts_lbs)
metrics.plot_roc_curve(svm_cls, vgg_ts_ft, ts_lbs)
plt.show()
```

Listing 8: SVM classification and evaluation.

Full Parameters Setting of *"svm.SVC()"*: Default

$classsklearn.\mathbf{svm.SVC}(*, \mathbf{C} = 1.0, \mathbf{kernel} =' rbf', \mathbf{degree} = 3,$
$\mathbf{coef0} = 0.0, \mathbf{shrinking} = True, \mathbf{probability} = False, \mathbf{tol} = 0.001,$
$\mathbf{gamma} =' scale', \mathbf{cache\_size} = 200, \mathbf{class\_weight} = None, \mathbf{verbose} = False,$
$\mathbf{max\_iter} = -1, \mathbf{decision\_function\_shape} =' ovr', \mathbf{break\_ties} = False,$
$\mathbf{random\_state} = None)$[7]

In this part of the code 8 we call a "**linear SVM**" classifier by **svm.SVC()** and simply use the **fit()** function to classify our training features, then the **predict()** function to get predictions about the test features and assign it to "`svm_lbs_pred`". We then use **metrics** from **sklearn** library to evaluate our classifier, we got excellent results (Accuracy: 0.97, Precision: 1.0, Recall: 0.7, F1-score: 0.82) (See figure 3.11).

```
Accuracy: 0.9719626168224299
Precision: 1.0
Recall: 0.7
F1-score: 0.8235294117647058
```

Figure 3.11: Output of code 8: SVM classification and evaluation.

[7] https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

Performance evaluation:

The **SVM** achieved a high accuracy of **97%**, in other words it predicted **97%** of the classes correct which means **"104 out of 107"** right predictions, with a precision of **1.0** which is the perfect score that means that SVM predicted all **positive identifications (True positive or Malignancy) as positive**, with a high **recall (also known as sensitivity)** of **0.7** which means that **false negatives** were little in other words the **benign images predicted as malignant** were little. The **F1-score** is simply the **mean (harmonic mean)** between **precision and recall**, which ever one get high the **F1-score** get higher, **SVM** has **0.84** means both precision and recall are high as we explained.

2.9 Classification with Multilayer Perceptron

```
# Call ANN classif
ann_cls = MLPClassifier(solver='adam', hidden_layer_sizes=(500,
 ↪  150), max_iter=20000)
# Training ANN
ann_cls.fit(vgg_tr_ft,tr_lbs)
# Testing ANN
ann_lbs_pred = ann_cls.predict(vgg_ts_ft)
# Print evaluation metrics
# Print evaluation metrics
print("Accuracy:",metrics.accuracy_score(ts_lbs, ann_lbs_pred))
print("Precision:",metrics.precision_score(ts_lbs, ann_lbs_pred))
print("Recall:",metrics.recall_score(ts_lbs, ann_lbs_pred))
print("F1-score:",metrics.f1_score(ts_lbs, ann_lbs_pred))
metrics.plot_confusion_matrix(ann_cls, vgg_ts_ft, ts_lbs)
metrics.plot_roc_curve(ann_cls, vgg_ts_ft, ts_lbs)
plt.show()
```

Listing 9: Multilayer Perceptron classification and evaluation.

Full Parameters Setting of *"MLPClassifier()"*: Customized

$classsklearn.neural\_network.$**MLPClassifier**(**hidden_layer_sizes** $= (500, 150)$ **activation** $=' relu', *,$ **solver** $=' adam',$ **alpha** $= 0.0001,$ **batch_size** $=' auto',$ **learning_rate** $=' constant',$ **learning_rate_init** $= 0.001,$ **power_t** $= 0.5,$ **max_iter** $= 20000,$ **shuffle** $= True,$ **random_state** $= None,$ **tol** $= 0.0001,$ **verbose** $= False,$ **warm_start** $= False,$ **momentum** $= 0.9,$ **nesterovs_momentum** $= True,$ **early_stopping** $= False,$ **validation_fraction** $= 0.1,$ **beta_1** $= 0.9,$ **beta_2** $= 0.999,$ **epsilon** $= 1e-08,$ **n_iter_no_change** $= 10,$ **max_fun** $= 15000)$[8]

[8] https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html

In this code 9, we train a second classifier, the "**Multilayer Perceptron**" by calling **MLPClassifier** with some specific parameters like we set **Adam as The solver for weight optimization**. We train and test the "neural network" classifier applying the same functions, **fit()** on the training features and **predict()** on test ones, the we save the predicted classes as "`ann_lbs_pred`".
We use the same **metrics** as last time for evaluation, we got excellent results again (Accuracy: 0.96, Precision: 0.75, Recall: 0.9, F1-score: 0.82) (See figure 3.12).

```
Accuracy: 0.9626168224299065
Precision: 0.75
Recall: 0.9
F1-score: 0.8181818181818182
```

Figure 3.12: Output of code 9: Multilayer Perceptron classification and evaluation.

Performance evaluation:

The **MLP** achieved a high accuracy of **96%** so it predicted **96%** of the classes correct, approximately **"103 out of 107"** right predictions, MLP achieved a precision of **0.75** which means that **75% of positive identifications (class=1)** were **correct** which also means that it can detect "**Malignancy**" with a **75% rate**. A high **recall (also known as sensitivity)** of **0.9** was achieved meaning that there were almost no **false negatives** or false malignancy prediction. The **MLP** has **0.81** as **F1-score** so both precision and recall are quite high which is a **good classifier** sign.

2.10 Classification with Random Forest

```
# Call RF classif
rf_cls = RandomForestClassifier()
# Training RF
rf_cls.fit(vgg_tr_ft,tr_lbs)
RandomForestClassifier(...)
# Testing RF
rf_lbs_pred = rf_cls.predict(vgg_ts_ft)
# Print evaluation metrics
print("Accuracy:",metrics.accuracy_score(ts_lbs, rf_lbs_pred))
print("Precision:",metrics.precision_score(ts_lbs, rf_lbs_pred))
print("Recall:",metrics.recall_score(ts_lbs, rf_lbs_pred))
print("F1-score:",metrics.f1_score(ts_lbs, rf_lbs_pred))
metrics.plot_confusion_matrix(rf_cls, vgg_ts_ft, ts_lbs)
metrics.plot_roc_curve(rf_cls, vgg_ts_ft, ts_lbs)
plt.show()
```

Listing 10: Random forest classification and evaluation.

Full Parameters Setting of *"RandomForestClassifier()"*: Default

$classsklearn.ensemble.\mathbf{RandomForestClassifier}(\mathbf{n\_estimators} = 100, *,$
$\mathbf{criterion} =' gini', \mathbf{max\_depth} = None, \mathbf{min\_samples\_split} = 2,$
$\mathbf{min\_samples\_leaf} = 1, \mathbf{min\_weight\_fraction\_leaf} = 0.0, \mathbf{max\_features} =' auto',$
$\mathbf{max\_leaf\_nodes} = None, \mathbf{min\_impurity\_decrease} = 0.0,$
$\mathbf{min\_impurity\_split} = None, \mathbf{bootstrap} = True, \mathbf{oob\_score} = False,$
$\mathbf{n\_jobs} = None, \mathbf{random\_state} = None, \mathbf{verbose} = 0, \mathbf{warm\_start} = False,$
$\mathbf{class\_weight} = None, \mathbf{ccp\_alpha} = 0.0, \mathbf{max\_samples} = None)$[9]

In this code 10, we train another well-known classifier "**Random forest**", we call it with the **RandomForestClassifier()**, then train it and evaluate it with **fit()** and **predict()**, we save the predicted classes as "`rf_lbs_pred`".
The same **evaluation metrics** are used, the RF classifier got the following results (Accuracy: 0.96, Precision: 0.75, Recall: 0.9, F1-score: 0.82) (See figure 3.12).

```
Accuracy: 0.9532710280373832
Precision: 0.6923076923076923
Recall: 0.9
F1-score: 0.7826086956521738
```

Figure 3.13: Output of code 10: Random forest classification and evaluation.

Performance evaluation:

The **RF** has a **95%** of accuracy so it predicted **95%** correct, approximately **"102 out of 107"** right predictions, the precision was fairly good with **69%** which means that **69% of positive identifications (class=1)** were **correct** which also means that it can detect "**Malignancy**" with a **69% rate**. A high **recall (also known as sensitivity)** of **0.9** was achieved meaning that there were almost no **false negatives** or false malignancy prediction. The **RF** has **0.78** as **F1-score** because the recall is high.

[9] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

3 Synthesis discussion:

We split this discussion into two sub-parts, the first one to give a global discussion of the results of each classifier in contrast to the other, the second one is to give a fair comparison between the original study by *Li Shen* and the results of ours.

3.1 Part 01: Results interpretation

The main purpose of this study is to present a modern image classification approach and prove its advantage taking account of what has been stated in the *State-of-the-art* Chapter.

First, we have used a ***VGG-16*** *as features extractor* (see code 3) by applying the weights from the ***"global_average_pooling2d_1"*** layer of *Li Shen*'s model on a *training set* (280 images) (see code 4) and a *test set* (107 images) (see code 5).
The extracted features vectors ***"vgg_tr_ft & vgg_ts_ft"*** along with the labels vectors ***"tr_lbs & ts_lbs"*** (see code 6) are used for the classification process with three different classifiers: ***SVM***, ***MLP*** and ***RF***.

In the following table, we gather all results obtained with the ***SVM***, ***MLP*** and ***RF*** classifiers for comparison purpose:

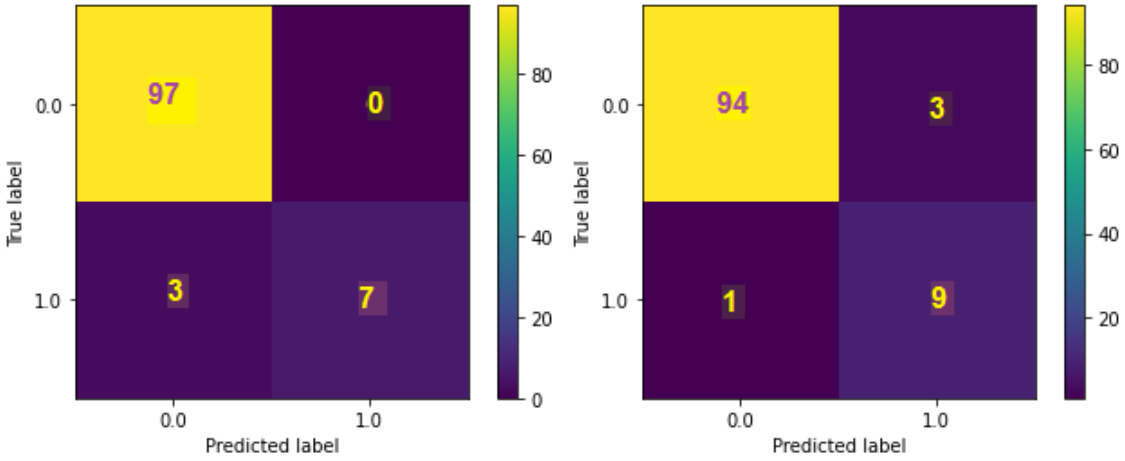
Classif.	Accuracy	Precision	Recall	F1-Score
SVM	0.97	1.0	0.7	0.82
MLP	0.96	0.75	0.9	0.82
RF	0.95	0.69	0.9	0.78

Table 3.1: Classification Results: ACC, Precision, Recall & F1-score.

all of them gave excellent close results as it's figured in the table 3.1, from which we conclude that ***SVM*** scored the highest accuracy ***"ACC=0.97"*** and a perfect precision ***"Precision=1.0"*** which means that it was able to classify *(104/107) images correctly* identifying *all benign images as negatives (True Negative) so* ***"TN=97 or 97 images as Class=0"*** (see figure 3.14a).
With that being said, ***SVM*** had an advantage on *the overall performance*, compared to both ***MLP*** and ***RF*** with ***"ACC=0.96/ 0.95"*** and ***"Precision=0.75/ 0.69"*** respectively (see table 3.1) which means that some *benign images were classified as positive* (False Positive) as it is figured in the confusion matrices ***"MLP: FP=3"*** (see figure 3.14b) & ***"RF: FP=4"*** (see figure 3.15), such result only raise concern to the patient with zero risk.

On the other hand, what could be a risk is when *malignant images are classified as negatives* (False Negative) for which we have ***"SVM: FN=3"*** (see figure 3.14a) and ***"MLP & RF: FN=1"*** (see figure 3.14b & 3.15) which is represented in the *Recall* scores ***"SVM: Recall=0.7"*** and ***"MLP & RF: Recall=0.9"*** (see table 3.1).
That being said, ***MLP*** & ***RF*** are better in *classifying malignancy* than ***SVM***.

Lastly, the ***F1-Score*** combines both *Precision* and *Recall* to give more global evaluation for which ***SVM*** & ***MLP*** are equivalent with *"**F1-Score=0.82**"* for both (see table 3.1).
So judging on the *"Accuracy"* and *"F1-Score"* at the same time, we conclude from the table 3.1 that ***SVM*** (ACC=0.97 & F1-Score=0.82) and ***MLP*** (ACC=0.96 & F1-Score=0.82) barely outperform one another but surely both of them outperform ***RF***, so we see that ***SVM*** and ***MLP*** are *equivalently good performing* in our case.



(a) SVM (b) MLP

Figure 3.14: Confusion matrix for SVM and MLP.

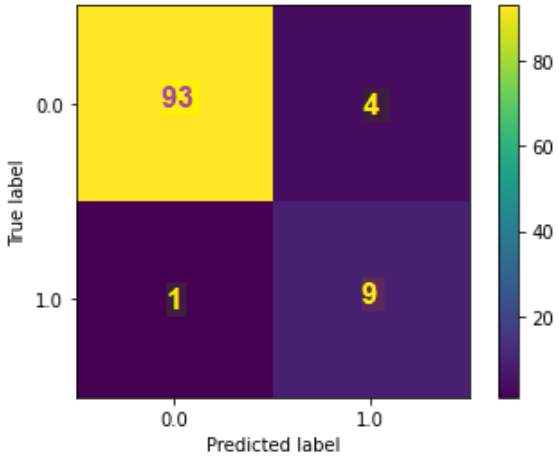


Figure 3.15: Confusion matrix for RF.

3.2 Part 02: Fair Comparison with Li Shen's Study

As our purpose is not coming up with a new architecture nor achieving a new state-of-the-art record, and as we have made it clear introducing the first part about the purpose of this study, we prefer to cut the road of all possible unnecessary struggles that would be implicated along the process by using a successful study by *Li Shen*'s team in the same area of interest as our which is *"Classification*

of Mammograms with CNN".

Since we have used *the exact same network as as* ***Mr. Li*** *excluding only the last dense layer "****dense_1****"* that is a fully connected layer which plays a role of a classifier by calculating the probability of each class, and eventually replace it with a third-party classifier (***SVM***, ***MLP*** and ***RF***) to explore new insights on the performance of third-party classifier in a hybrid setting with a ***VGG-16*** as features extractor on medical images.
Hence any successfully achieved results are attributed in first place to *Li Shen*'s architecture and configuration.

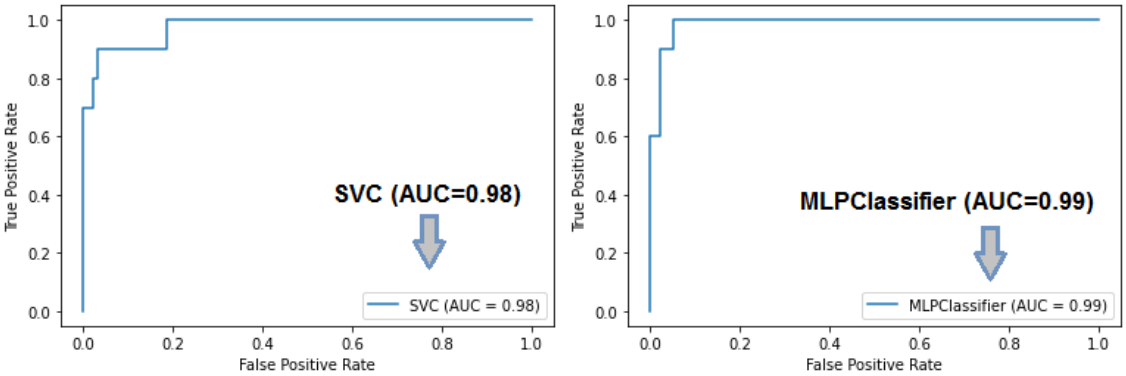
To give a loyal and fair comparison we must use the same metrics, for that we use the *AUC score* as it's the only measurement given in *Li Shen*'s study for the *"whole image classifier"* and their results are stated in their *GitHub Repository manual "README.md"* [59] as well as their paper [58].
In the table 3.2 we can see all *AUC scores* of *ROC curves* for *Li shen end-to-end model* ***(VGG-16: AUC=0.96)*** as well as our classifiers ***(SVM: AUC=0.98, MLP: AUC=0.99, RF: AUC=0.99)*** (see the *ROC curves* for ***SVM*** 3.16a, ***MLP*** 3.16b and ***RF*** 3.17).

The *AUC (Area under the Curve)* is a measurement used in classification analysis in order to determine which of the used models predicts the classes best, usually used with *ROC curves* where the *true positive rates* (aka. *Sensitivity* or *Recall*) $TPR = \frac{TP}{TP+FN}$ are plotted against *false positive rates* $FPR = \frac{FP}{FP+TN}$ [61].
So having a higher *AUC* means the model is more accurate at identifying the positive subjects (or the malignancy) which is a critical feature.
Now laying on the earlier discussed results in the ***"Part 01"***, we see that ***SVM*** with ***"AUC=0.98"*** had ***"FN=3"*** and ***MLP & RF*** both with ***"AUC=0.99"*** had ***"FN=1"***, so ***0.01*** difference in *AUC score* means ***2*** more *malignant images* correctly classified therefore ***MLP*** and ***RF*** achieved ***0.03*** difference in *AUC* over the ***VGG-16 (AUC=0.96)*** which is even worth ***6*** more *malignant (positive)* subjects correctly classified, so it is a considerable improvement fairly saying.

Clearly our classifiers gave a slightly higher *AUC score* than the *end-to-end* ***VGG-16*** which certainly reflects the advantage of a third-party classifier performance like ***SVM***, ***MLP*** and ***RF*** over the ***dense*** *fully connected last layer* employed in *Li Shen*'s model because we have used the same dataset and the same ***CNN*** network configuration.

Classif.	**AUC**
SVM	0.98
MLP	0.99
RF	0.99
VGG-16	**0.96**

Table 3.2: Classification Results: AUC.



(a) SVM (b) MLP

Figure 3.16: ROC curves for SVM and MLP.

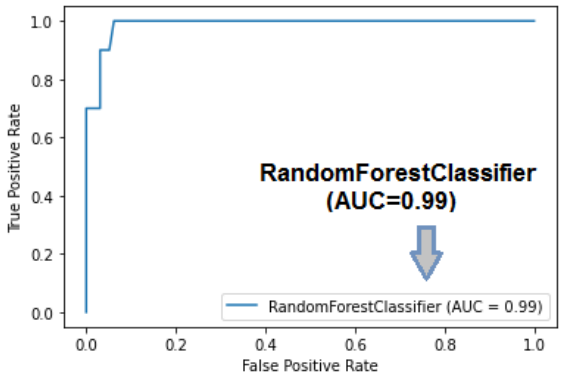


Figure 3.17: ROC curve for RF.

4 Conclusion

In this chapter, we have conducted the features extraction based classification experiments with *Li Shen*'s *VGG-16* network and three different classifiers (*SVM*, *MLP* and *RF*) in three different hybrid settings each time.

At the end, we have concluded that all *three classifiers* gave excellent results and slightly better *AUC* score than the *end-to-end approach*, specially the *MLP* and *RF*.

So the proposed approach was successful regarding *INbreast mammograms classification* for two reasons, first is the fine-tuned *VGG-16* parametrized architecture by *Li Shen* that could be able to extract useful features, and second for the third-party classifiers that took most advantage of these features especially the *MLP* and *RF* as a replacement for the *dense layer*. Hence a hybrid setting of *VGG-16* and *MLP* or *RF* is recommended for *"Breast Mammograms classification"* tasks.

General conclusion and perspectives

In this work, we have introduced a newer and recent approach for image classification in which meaningful features are automatically extracted from the images with a Convolutional Neural Network (CNN).

We tested this study on the INbreast dataset, with extracted and transferred features from a VGG-16 pre-trained on similar data which is in our case study "Breast Cancer Mammograms", we fine-tuned these features again with the same CNN architecture (VGG-16) but we replaced the last layer with a non-linear classifier for which we tried three of the most popular: "Support vector machines (SVM)", Multilayer perceptron (MLP, also known as neural networks)", and "Random forests (RF)".

The results revealed a very good performance and even an enhancement with SVM instead of the end-to-end approach, this enhancement can give an advantage, especially in such a sensitive domain as healthcare, and with a big mass of data, the marginal enhancement can increase.

Our work presents two main points:

- The CNN-based approach for medical image classification as a new and advantageous prospect
- Including a hybrid setting along with CNN can give better results depending on the problem treated and the data in hand

In the comparison of SVM to RF and MLP, last two can be better at classifying the malignancy on INbreast dataset than SVM which we can be taking advantage of in a more complex model that includes all three classifiers.

Perspectives

CNN deep features are increasingly employed with different settings and architectures on different application domains, one of the best performing approaches, as we have seen in the state-of-the-art, is the fusing of classification scores from multiple hybrid classifiers, which open new prospects for future studies in which we can follow such model setting to fuse classification scores of all of SVM, ANN, and RF for example to get further improvements.

Also, we should know that the VGG-16 is not the best performing CNN architecture in the state-of-the-art as many international large-scale competitions and challenges discovered far more complex and extremely useful architectures such as ResNet-50, DenseNet, and GoogLeNet, hence we can consider them as well in a fine-tuned setting to extract more accurate features.

Bibliography

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

[2] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[3] F. Chollet, *Deep Learning mit Python und Keras: Das Praxis-Handbuch vom Entwickler der Keras-Bibliothek*. MITP-Verlags GmbH & Co. KG, 2018.

[4] J. D. Kelleher, *Deep learning*. Mit Press, 2019.

[5] C. C. Aggarwal, *Neural networks and deep learning*. Springer, 2018.

[6] Y. LeCun *et al.*, "Generalization and network design strategies," *Connectionism in perspective*, vol. 19, no. 143-155, p. 18, 1989.

[7] M. Eickenberg, A. Gramfort, G. Varoquaux, and B. Thirion, "Seeing it all: Convolutional network layers map the function of the human visual system," *NeuroImage*, vol. 152, pp. 184–194, 2017.

[8] J. Patterson and A. Gibson, *Deep learning: A practitioner's approach*. " O'Reilly Media, Inc.", 2017.

[9] D. Purves, G. J. Augustine, D. Fitzpatrick, *et al.*, "Neuroscience. 4th," *Sunderland, Mass.: Sinauer. xvii*, vol. 857, p. 944, 2008.

[10] D. Purves, *Brains: how they seem to work*. Ft Press, 2010.

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

[12] C. C. Aggarwal, *Data mining: the textbook*. Springer, 2015.

[13] K. Fukushima, "Recent advances in the deep cnn neocognitron," *Nonlinear Theory and Its Applications, IEICE*, vol. 10, no. 4, pp. 304–321, 2019.

[14] A. S. Lundervold and A. Lundervold, "An overview of deep learning in medical imaging focusing on mri," *Zeitschrift für Medizinische Physik*, vol. 29, no. 2, pp. 102–127, 2019.

[15] A. Ghatak, *Deep learning with R*. Springer, 2019, vol. 245.

[16] A. Rosebrock, *Deep Learning for Computer Vision with Python: Starter Bundle*. PyImageSearch, 2017.

[17] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

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

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

[20] C. Szegedy, W. Liu, Y. Jia, *et al.*, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

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

[22] Q. Yang, Y. Zhang, W. Dai, and S. J. Pan, *Transfer learning*. Cambridge University Press, 2020.

[23] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[24] L. Torrey and J. Shavlik, "Transfer learning," in *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, IGI global, 2010, pp. 242–264.

[25] M. Sonka, V. Hlavac, and R. Boyle, *Image processing, analysis, and machine vision*, Fourth edition. Cengage Learning, 2014.

[26] S. J. Prince, *Computer vision: models, learning, and inference*. Cambridge University Press, 2012.

[27] R. Szeliski, *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[28] N. Snavely, S. M. Seitz, and R. Szeliski, "Photo tourism: Exploring photo collections in 3d," in *ACM siggraph 2006 papers*, 2006, pp. 835–846.

[29] M. Goesele, N. Snavely, B. Curless, H. Hoppe, and S. M. Seitz, "Multi-view stereo for community photo collections," in *2007 IEEE 11th International Conference on Computer Vision*, IEEE, 2007, pp. 1–8.

[30] H. Sidenbladh, M. J. Black, and D. J. Fleet, "Stochastic tracking of 3d human figures using 2d image motion," in *European conference on computer vision*, Springer, 2000, pp. 702–718.

[31] J. Sivic, C. L. Zitnick, and R. Szeliski, "Finding people in repeated shots of the same scene.," in *BMVC*, vol. 2, 2006, p. 3.

[32] C. K. Reddy and C. C. Aggarwal, *Healthcare data analytics*. CRC Press, 2015, vol. 36.

[33] T. G. Dietterich, R. H. Lathrop, and T. Lozano-Pérez, "Solving the multiple instance problem with axis-parallel rectangles," *Artificial intelligence*, vol. 89, no. 1-2, pp. 31–71, 1997.

[34] A. K. Sangaiah, *Deep Learning and Parallel Computing Environment for Bioengineering Systems*. Academic Press, 2019.

[35] J. Watt, R. Borhani, and A. Katsaggelos, *Machine learning refined: foundations, algorithms, and applications*. Cambridge University Press, 2016.

[36] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[37] G. Dong and H. Liu, *Feature engineering for machine learning and data analytics*. CRC Press, 2018.

[38] M. Nixon and A. Aguado, *Feature extraction and image processing for computer vision*. Academic press, 2019.

[39] A. Zheng and A. Casari, *Feature engineering for machine learning: principles and techniques for data scientists*. " O'Reilly Media, Inc.", 2018.

[40] T. Malisiewicz, "From feature descriptors to deep learning: 20 years of computer vision," *Tombone's Computer Vision Blog*, Jan. 2015. [Online]. Available: https://www.computervisionblog.com/2015/01/from-feature-descriptors-to-deep.html.

[41] R. Venkatesan and B. Li, *Convolutional neural networks in visual computing: a concise guide*. CRC Press, 2017.

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

[43] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "Cnn features off-the-shelf: An astounding baseline for recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2014, pp. 806–813.

[44] J. Amin, M. Sharif, M. Yasmin, T. Saba, M. A. Anjum, and S. L. Fernandes, "A new approach for brain tumor segmentation and classification based on score level fusion using transfer learning," *Journal of medical systems*, vol. 43, no. 11, p. 326, 2019.

[45] S. Basheera and M. S. S. Ram, "Classification of brain tumors using deep features extracted using cnn," in *Journal of Physics: Conference Series*, IOP Publishing, vol. 1172, 2019, p. 012 016.

[46] Z. Jiao, X. Gao, Y. Wang, and J. Li, "A deep feature based framework for breast masses classification," *Neurocomputing*, vol. 197, pp. 221–231, 2016.

[47] A.-A. Nahid and Y. Kong, "Local and global feature utilization for breast image classification by convolutional neural network," in *2017 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, IEEE, 2017, pp. 1–6.

[48] F. A. Spanhol, L. S. Oliveira, P. R. Cavalin, C. Petitjean, and L. Heutte, "Deep features for breast cancer histopathological image classification," in *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, IEEE, 2017, pp. 1868–1873.

[49] Z. Cao, L. Duan, G. Yang, T. Yue, and Q. Chen, "An experimental study on breast lesion detection and classification from ultrasound images using deep learning architectures," *BMC medical imaging*, vol. 19, no. 1, p. 51, 2019.

[50] Y. Xie, J. Zhang, S. Liu, W. Cai, and Y. Xia, "Lung nodule classification by jointly using visual descriptors and deep features," in *Medical Computer Vision and Bayesian and Graphical Models for Biomedical Imaging*, Springer, 2016, pp. 116–125.

[51] C. Wang, A. Elazab, J. Wu, and Q. Hu, "Lung nodule classification using deep feature fusion in chest radiography," *Computerized Medical Imaging and Graphics*, vol. 57, pp. 10–18, 2017.

[52] A. Teramoto, T. Tsukamoto, Y. Kiriyama, and H. Fujita, "Automated classification of lung cancer types from cytological images using deep convolutional neural networks," *BioMed research international*, vol. 2017, 2017.

[53] A. Nibali, Z. He, and D. Wollersheim, "Pulmonary nodule classification with deep residual networks," *International journal of computer assisted radiology and surgery*, vol. 12, no. 10, pp. 1799–1808, 2017.

[54] B. Harangi, A. Baran, and A. Hajdu, "Classification of skin lesions using an ensemble of deep neural networks," in *2018 40th annual international conference of the IEEE engineering in medicine and biology society (EMBC)*, IEEE, 2018, pp. 2575–2578.

[55] Z. Yu, D. Ni, S. Chen, *et al.*, "Hybrid dermoscopy image classification framework based on deep convolutional neural network and fisher vector," in *2017 IEEE 14th International Symposium on Biomedical Imaging (ISBI 2017)*, IEEE, 2017, pp. 301–304.

[56] A. Mahbod, G. Schaefer, C. Wang, R. Ecker, and I. Ellinge, "Skin lesion classification using hybrid deep neural networks," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2019, pp. 1229–1233.

[57] I. C. Moreira, I. Amaral, I. Domingues, A. Cardoso, M. J. Cardoso, and J. S. Cardoso, "Inbreast: Toward a full-field digital mammographic database," *Academic radiology*, vol. 19, no. 2, pp. 236–248, 2012.

[58] L. Shen, L. R. Margolies, J. H. Rothstein, E. Fluder, R. B. McBride, and W. Sieh, "Deep learning to improve breast cancer early detection on screening mammography," *arXiv preprint arXiv:1708.09427*, 2017.

[59] L. Shen, *End2end-all-conv*, https://github.com/lishen/end2end-all-conv, 2016.

[60] E. Mendelson, J. Baum, W. Berg, C. Merritt, and E. Rubin, "Breast imaging reporting and data system: Acr bi-rads—breast imaging atlas," *BI-RADS: Ultrasound Reston, American College of Radiology, VA*, 2003.

[61] *What does auc stand for and what is it?* https://stats.stackexchange.com/questions/132777/what-does-auc-stand-for-and-what-is-it.

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

[63] A. C. Müller, S. Guido, *et al.*, *Introduction to machine learning with Python: a guide for data scientists*. " O'Reilly Media, Inc.", 2016.

[64] A. Myers, "Stanford's john mccarthy, seminal figure of artificial intelligence, dies at 84," Oct. 2011. [Online]. Available: https://news.stanford.edu/news/2011/october/john-mccarthy-obit-102511.html.

[65] Y. LeCun, "Predictive learning," Neural Information Processing Systems Conference - NIPS 2016, Jan. 2017. [Online]. Available: https://channel9.msdn.com/Events/Neural-Information-Processing-Systems-Conference/Neural-Information-Processing-Systems-Conference-NIPS-2016/Predictive-Learning.

[66] T. Peng, "Yann lecun cake analogy 2.0," *Medium*, M. Sarazen, Ed., Feb. 2019. [Online]. Available: https://medium.com/syncedreview/yann-lecun-cake-analogy-2-0-a361da560dae.

[67] A. Burkov, *The hundred-page machine learning book*. Andriy Burkov Quebec City, Can., 2019, vol. 1.

[68] D. Grattarola, "Deep feature extraction for sample-efficient reinforcement learning," M.S. thesis, Politecnico Di Milano, Oct. 2017.

[69] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*, Second edition. MIT press, 2018.

[70] P. Wilmott, *Machine learning: an applied mathematics introduction*, First edition. Panda Ohana Publishing, 2019.

[71] I. Vasilev, D. Slater, G. Spacagna, P. Roelants, and V. Zocca, *Python Deep Learning: Exploring deep learning techniques and neural network architectures with Pytorch, Keras, and TensorFlow*, Second edition. Packt Publishing Ltd, 2019.

[72] Z. SALLOUM, "Math behind reinforcement learning, the easy way," *Towards Data Science*, Aug. 2018. [Online]. Available: https://towardsdatascience.com/math-behind-reinforcement-learning-the-easy-way-1b7ed0c030f4.

[73] E. S. Olivas, J. D. M. Guerrero, M. Martinez-Sober, J. R. Magdalena-Benedito, L. Serrano, *et al.*, *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques: Algorithms, Methods, and Techniques*. IGI Global, 2009.

[74] W. Burger and M. J. Burge, *Digital image processing: an algorithmic introduction using Java*. Springer, 2016.

[75] E. R. Davies, *Computer vision: principles, algorithms, applications, learning*. Academic Press, 2017.

[76] D. A. Forsyth and J. Ponce, *Computer vision: a modern approach*. Prentice-Hall, 2012.

[77] A. Ng, "Machine learning and ai via brain simulations," 2013. [Online]. Available: http://datascienceassn.org/sites/default/files/Machine%20Learning%20and%20AI%20via%20Brain%20Simulations.pdf.