الجمهورية الجزائرية الديمقراطية الشعبية

People's Democratic Republic of Algeria

Ministry of Higher Education and Scientific Research

University of May 8th, 1945 - Guelma -

Faculty of Mathematics, Computer Science and Material Sciences

Computer Science Department

Master's Thesis

Sector: IT

Option: STIC

Theme:

CNN Based Deep Face Recognition

Supervisor:

Mr. Hallaci Samir

Student:

Randa Aouassa

October 2020

Acknowledgment

بسم الله الرحمان الرحيم و الحمد لله

I would like to thank my dear teacher and supervisor Mr.Hallaci for supporting me throughout this tough year who kept motivating me to give the best I could regarding this covid-19 pandemic.

A kind thanks to the jury members for giving me the honor of judging my work. I also deeply thank all the computer science department teachers

Finally, I thank my family, friends, colleagues and all the class of 2020

Dedication

بسم الله الرحمان الرحيم و الحمد لله

I dedicate this work to

My dearest Brother whom without my life is meaningless

My parents, who raised, teached, encouraged and supported me throughout all my life

All my bigger family members, my friends and colleagues

"Success is no accident it is hard work, perseverance,
Learning, Studying, Sacrificing, And most of all, Love
Of what you are doing or learning to do."

Abstract

In this study, we presented the newest approaches used for facial detection and recognition based on deep learning and specifically convolutional neural networks.

In our project, we plan to use a deep convolutional neural network (CNN) to extract the characteristics of the input images, and to recognize faces in a complex environment based on one of the most accurate architectures (VGG).we focused on transfer learning to make use of the existing models that have been already trained on massive datasets, such as VGG16 for facial recognition which is trained on the well known ImageNet dataset. We made use of the Dlib face detector that also utilize deep learning technology to extract the face's coordinates. We created our own dataset which is a collection of images from the web, and we focused on making the dataset richer by applying data augmentation technology (rotation, shifting...) that helped to create newer set of images from the existing ones. The performance of our facial recognition system was evaluated on our own custom dataset as well as random unknown images from the web and it achieved impressive results.

Key words:

Face Recognition, Face detection, Deep learning, Convolutional neural networks, Models, Datasets, Tensorflow, Keras

Table of Contents

List of Figures:

List of Tables:

General Introduction

Face recognition in images or videos is one of the biggest challenges in computer vision and pattern recognition. The problem and the main difficulty comes from variations in the appearance of the face caused by factors such as expression, lighting, partial occlusion of the face. Briefly, a face recognition system extracts the characteristics of an input image and compares them with the characteristics of labeled faces from a database. The comparison is based on a characteristic similarity metric and the label of the most similar database entry is used to label the input image.

This technology has seen wider uses in recent years on mobile platforms and in other forms of technology, such as robotics. It is typically used as access control in security systems and can be compared to other biometrics such as fingerprint or eye iris recognition systems. Although the accuracy of facial recognition system as a biometric technology is lower than iris recognition or fingerprint recognition, it is still widely adopted due to its contactless and non-invasive process. Recently, it has also become popular as a commercial identification and marketing tool. Other applications include advanced human-computer interaction, video surveillance, automatic indexing of images, and video database, among others.

For the requested work, we plan to use a deep convolutional neural network (CNN) to extract the characteristics of the input images, and to recognize faces in a complex environment based on one of the most accurate architectures (VGG).

The Tensorflow architecture across the Keras backend will be used to implement the deep neural network (CNN) algorithm. Moreover, in order to align the faces on the input images we will use Dlib and OpenCV. The performance of our facial recognition system will be evaluated on our own custom dataset.

The objectives of the system will be broken down into three main stages:

1. Detect, transform and crop faces from the input images. This ensures that the faces are aligned before inserting them into the CNN.
2. Use CNN to extract representations in reduced dimensions, or incorporations of faces from aligned input images to build vectors. In this space, the distance corresponds directly to a measure of the similarity of the faces.
3. Compare the input embedding vectors to the labeled embedding vectors in the database.

Thesis Organization:

We chose to articulate our study in three main chapters:

- **Chapter 01: Deep Face Recognition Fundamentals**

This chapter aims at a detailed analysis of different approaches and techniques used for detection and face recognition in the field of Deep Learning.

Part 01: Fundamentals of Face Detection and recognition

Part 02: History and details on artificial intelligence technology focusing on its main subset Deep Learning.

Part 03: State of the art models and State of the art datasets.

- **Chapter 02: Convolutional Neural Network**

In this chapter, we explained everything there is to know about CNN starting from its basic building block to its implementation.

- **Chapter 03 : Conception, Implementation and results**

In this chapter, we detail the different system steps that we have elaborate. In addition, we present the experimental results obtained.

Finishing up with a general conclusion, to summarize our contributions, and giving some perspective on future work.

I. Deep Face Recognition Fundamentals

1. Introduction:

Face Recognition is definitely one of the most popular computer vision problems, and is one of the many wonders that AI research has brought forward to the world. It is a subject of curiosity for many researchers. Thanks to its popularity it has been well studied over the last 50 years, so we can define it as the problem of identifying and verifying people in a photograph by their face.

It is a task that is trivially performed by humans, even under complex conditions, such as varying light and when faces are changed by age or obstructed with accessories and facial hair. Nevertheless, it is remained a challenging computer vision problem for decades.

Deep learning methods are able to leverage very large datasets of faces and learn rich and compact representations of faces, allowing modern models to first perform as well and later to outperform the face recognition capabilities of humans.

To understand how a machine can recognize faces, we can start with asking ourselves — how do we recognize a face? Most images of human faces have two eyes, a nose, lips, forehead, chin, ears, hair… That rarely changes. Yet, faces are different from each other. What makes them different? At the same time, face of the same person changes with emotion, expression, age… In fact just change in orientation creates a different image. How do we identify a person in spite of all that?

In this Chapter, we will discover the problem of face recognition and how deep learning methods can achieve superhuman performance, and for that we are going to present:

- The fundamentals of face detection and recognition
- The history and evolution of Artificial Intelligence, Machine Learning and Deep Learning
- A basic explanation of AI and ML, and a deeper focus on AI's main subset which is Deep Learning, starting from its basic building blocks to the last step which is choosing a dataset and training a model
- The latest and state of the art works applied in the field on Deep Face Recognition, with most used datasets and models.

2. Face Detection and Recognition Overview:

Faces are a complex multidimensional visual models and developing a computational model for face recognition is difficult. However, applying machine learning techniques have made it possible and with some real-time variations as well. Face recognition is a technology capable of identifying or verifying a person from a digital image or a video frame from a video source.

In order for the facial recognizing system to function, it is necessary to implement two main steps. First, it must detect the face within an image. Then, it must recognize that face from an existing database.

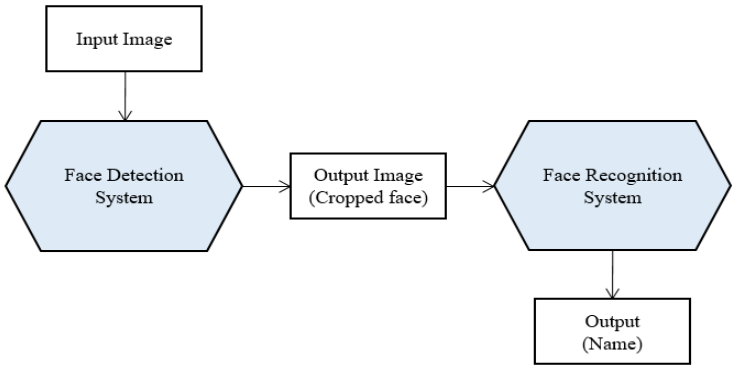


Figure 1.1: *Main Steps of Facial Recognition*

2.1. Face Detection:

- **Why detect a face**

Face detection cannot be automatic; therefore, it must be implemented accurately. An image might contain more than just a face, and in order for the system to function properly and recognize the faces precisely, it must separate each face candidate individually and label it, then pass it to the next step, which is the recognition.

- **How to detect a face**

Computers unlike us sees the world through combinations of numbers, an image is a collection of pixels, which represents the intensities of the colors in a form of a matrix, hence, it is only possible to understand an image through mathematical equations.

Scientists strive to build systems that can detect and recognize faces. One such system already exists: the human brain. In order to find the formation of a face in a mathematical equation, it needs to be broken into smaller functional problems taking reference on “how the brain classifies a face?”

A face is a summation of features or shapes (Eyes, eyebrows, mouth, nose, etc.) furthermore, the shapes are a set of connected edges and edges are identifiable using math [1], many approaches were proposed in order to solve these problems manually [2], but recently deep learning, which is based on what scientists think is the brain's structure of combined neurons, made an astonishing revolution in computer vision and many other fields subsidiary to artificial intelligence, unlike the standard methods, deep learning provides a new way of extracting and comparing features without the complete intervention of the human [3].

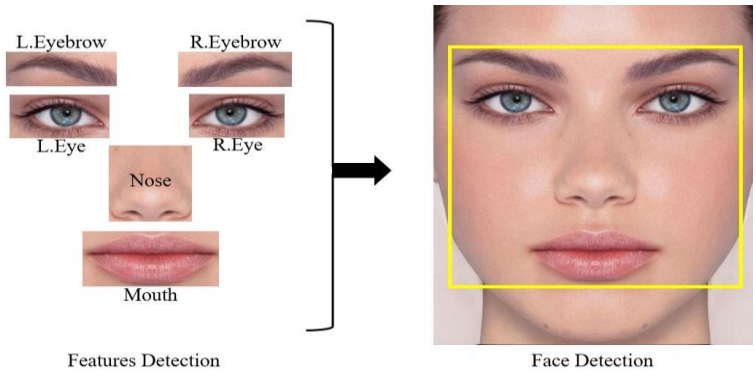


Figure 1.2: *Face Detection based on the facial features (eyes, nose...)*

- **The challenges in face detection**

Images of faces vary from one picture to another due to the alteration of different conditions from capturing equipment to lighting conditions, noises, etc. In face images, in addition to those challenges, we encounter the variation of expressions, poses, occlusions, etc. which are taken in consideration when building a face database for recognition [4].

2.2. Face Recognition:

- **Why recognize faces**

Contemplating the uniqueness of the human face, made Face recognition (FR) the eminent biometric technique and has been widely used in many more areas, such as military, finance, security and daily life in general.

- **How to recognize faces**

There are multiple methods in which facial recognition systems (FR) work [2], but in general, it is established through comparison of facial features from a given images with faces within a database. Initially, the process of face recognition was fulfilled in two steps. First, the effectuation of feature extraction and selection [5]. Second, classification [6][7]. However, with the revolution of deep learning and deep FR, the implementation of these two steps in details became unnecessary.

- **The challenges in face recognition**

The facial recognition system is yet to completely overcome the challenges which have constantly played with its quality of delivery, on the level of the image, illumination, background, pose, occlusion, expressions, complexity, etc. however, in the aspect of deep FR, the lack of saturated databases that affects hugely the learning process and therefore the recognition, fusion issues and the particular cases such as identicality (twins) and uncontrolled changes, and privacy-preserving concerns which are rising nowadays and made it difficult to collect data for study [8].

3. The Evolution of Artificial Intelligence:

3.1. The 1st Period 1950-1970:

Trivial problem solving, no practicality, GOFAI – Good old-fashioned AI:

Year	Event
1942	The 3 Laws of Robotics by Isaac Asimov, Other sets of laws have been proposed by researchers since then
1950	The Turing Test proposed by Alan Turing
1952	The first self-learning game program
1956	Dartmouth Conference, first use terms of "Artificial Intelligence/ A.I"
1957	General Problem Solver (GPS) by Newell
1958	McCarthy developed LISP programming language
1959	- The MIT AI Lab (McCarthy and Minsky) - The term "Machine Learning" by Samuel
1961	- First Industrial Robot (Unimate) working at GM - "SAINT" the first expert system by slagle (MIT)
1964	"STUDENT" the first AI program which understands natural language
1965	"ELIZA" the first AI based Chatbot and expert system
1966	- "Shakey" the first locomotive and intelligent robot (SRI) - "MAC HACK" chess-playing program by Greenblatt, MIT
1968	"SHRDLU" an early natural language understanding computer program
1970	"WABOT-1" the first anthropomorphic robot (Waseda University)
1972	"Prolog" logic programming language
1973	"Lighthill Report" the poor progress report caused the "First AI winter" which is Reduced funding for AI research
1974	- "MYCIN" the first rule based AI expert system for medical diagnostics - The first autonomous vehicle, a mechanical "slider" (Stanford)

***Table 1.1**: First period of the Evolution of AI, ML and DL 1950-1970*

3.2. The 2nd period 1980-2000:

Researchers feeding machines with labeled data, Projects: ICOT – Japan ’82, MCC – US ’83, Alvey – UK ’84. And algorithms began to appear as parts of larger systems. AI solutions proved to be useful throughout the technology industry, such as data mining, industrial robotics...etc.

Year	Event
1980	- LISP based machines developed and marketed - “INTERNIST-1” The first Commercial Expert System
1986	A driverless van by Mercedes-Benz, with cameras and sensors
1988	- “Bayesian Network”, BNs or belief nets, in invented by Pearl - The chatbots, “Jabberwachy” and “Cleverbot” invented by Carpenter
1989	The first autonomous vehicle created by CMU using neural network
1993	“Polly” the tour guide robot, behavior-based robotics (MIT)
1997	IBM’s Deep Blue beats Gary Kasparov in chess
1998	“Furby” the first pet toy robot for children
1999	- “Kismet” emotional AI (MIT AI Lab) - “AIBO” introduced the first AI domestic robot by Sony
2000	“ASIMO” humanoid robot released by Honda
2002	“Roomba” autonomous robot vacuum is released by i-Robot
2004	- The first challenge for autonomous vehicles by DARPA - NASA rovers “Spirit” and “Opportunity” exploring Mars
2005	AI based recommendation engines
2006	“Machine Reading” unsupervised autonomous understanding of text
2007	- “ImageNet” visual database for object recognition software research - “CUDA” launched by NVIDIA, a parallel computing platform and programming interface
2009	- Self Driving Car build by Google, by 2014 it passed Nevada’s self-driving test - AI researchers discover GPU (Graphics Processing Unit) for DL

***Table 1.2**: Second period of the Evolution of AI, ML and DL 1980-2000*

3.3. The 3rd Period 2010-2020:

The age of machine learning, Computers acquire knowledge from data, not humans. Large tech companies invest in commercial applications of AI/ML

Date	Event
2010	- Democratize Data Access begins for Image Recognition - Narrative Science’s AI demonstrates ability to write reports
2011	- Apple released “Siri” - IBM’s “Watson” wins Jeopardy clash

2013	- "NEIL" by CMU, a semantic image analyzer ML system - "Vicarious" passes first Turing test – CAPTCHA
2014	- "Cortana" by Microsoft - "Alexa" by Amazon
2015	- "TensorFlow" by Google Brain, a ML Library (TPU) - "Open AI" open source initiative to develop AI benefit of all humanity
2016	- "Google Home" by Google - "Alpha Go" Google's Deepmind has defeated Go's N°1 champion - NVIDIA announces supercomputer for DL and AI - "Sophia" humanoid robot by Hanson Robotics, the first robot citizen - "PyTorch" Open source ML Library
2017	- The facebook AI research lab trained two chatbots to communicate with each other in order to learn how to negotiate; the chatbots diverged from human language and invented their own language to communicate with one another. - "Caffe" Open source DL framework.
2018	- "BERT" by Google, the first bidirectional unsupervised language representation - "Bixby" introduced by Samsung - Facebook detects faces and shares photos with friends to whom those photos belong - Alibaba language processing AI outscored human intellect at a Standford reading and comprehension test.
2020	- DeepMind team uses DL algorithms "Agent 57" that outperforms humans at Atari games with Deep Reinforcement Learning. - Widespread "5G" network deployments worldwide

***Table 1.3**: Third period of the Evolution of AI, ML and DL 2010-2020*

4. Deep Face Recognition

Convolutional Neural Network is one of Deep learning architectures used in the facial recognition systems and computer vision in general, DL is subclass of machine learning which is a subclass of artificial intelligence, hence, to understand deep learning, it is necessary to start with the main fundamentals of AI and ML.

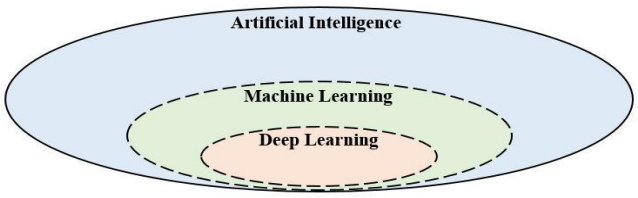


***Figure 1.3**: Artificial Intelligence's Subclasses*

4.1. Artificial Intelligence:

Artificial intelligence is the effort to automate intellectual tasks normally performed by humans. It starts with us identifying a set of rules and code it, the computer than execute this set of rules and follow these instructions, more rules resulted better AI, and eventually, AI was fully relied on human intervention with lots of programming where some programs were nearly half million lines of code and it can be either simple or complex.

4.2. Machine Learning:

British computer scientist Alan Turing and his team, created the first machine to decipher Enigma. His invention laid the foundations for Machine Learning. Machine learning is a study of computer algorithms that improve automatically through experience [9]. It made programming easier by taking in just training data as inputs, which is combined of pairs of data (features) and what their outputs should be (labels). ML figures out the patterns or the rules by itself, but the main goal in ML is to raise the accuracy as high as possible which means it still can make mistakes. However, to raise the accuracy, ML needs a massive amount of inputs as examples to train a good model that than predict the best possible output for new unknown data.

Tom M. [10] provided a widely quoted, more formal definition of the algorithms studied in the machine-learning field:

> "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E."

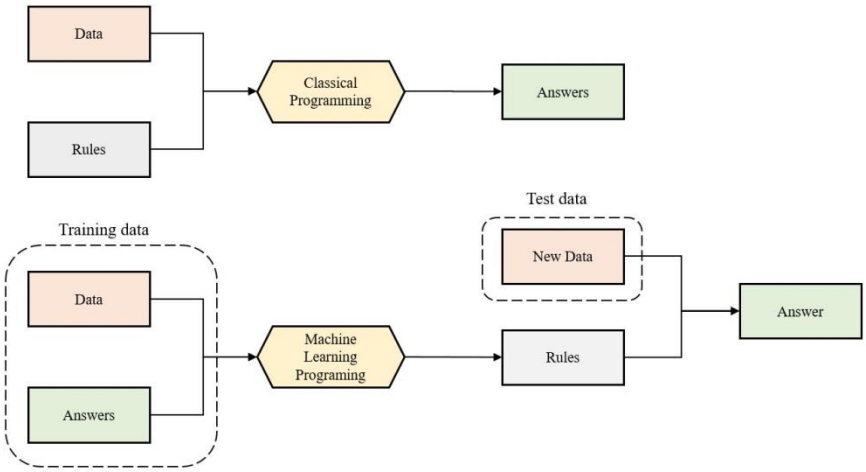


***Figure 1.4**: Difference between Classical Programming and ML Programming learning steps*

4.2.1. Machine Learning Data:

The data in machine learning is a set of inputs presented as a heterogeneous matrix, Rows of the matrix are called *observations, examples*, or *instances*, each one contain a set of measurements for a subject. Columns of the matrix are called *predictors, attributes,* or *features,* each are variables representing a measurement taken on every subject. The response data is a column vector where each row contains the output of the corresponding observation in the input data. To *fit* or to *train* a supervised learning model we must choose an appropriate algorithm, and then pass this training data to it.

The data types can be either *numeric vector, categorical vector, Character array, String array, Cell array of character vector,* or *logical vector.* For regression, the response data must be a numeric vector with the same number of elements as the number of rows of instances. While for classification, it can be any of the mentioned types above.

4.2.2. Machine Learning approaches:

There are three main machine learning approaches used nowadays [11]:

1) Supervised Learning:

Supervised learning is the most applicable type of learning; its algorithms build a mathematical model of a set of data that contains both the inputs and their desired outputs [12]. Called the Training data, where each training example is represented by an array or vector, sometimes called a feature vector, and the full training data is represented by a matrix.

Through iterative optimization of an objective function, supervised learning algorithms learn an inferred function that can be used to predict the output associated with new unknown inputs [13]. An optimal scenario will allow for the algorithm to correctly determine the class labels for unseen instances. This requires the learning algorithm to generalize from the training data to unseen situations in a “reasonable” way. The computer “learns” from the observations, when exposed to more observations, the computer than improves its predictive performance.

Specifically, a supervised learning algorithm takes a known set of input data and known responses to the data (output), and trains a model to generate reasonable predictions for the response to new data. (***figure 1.5***).

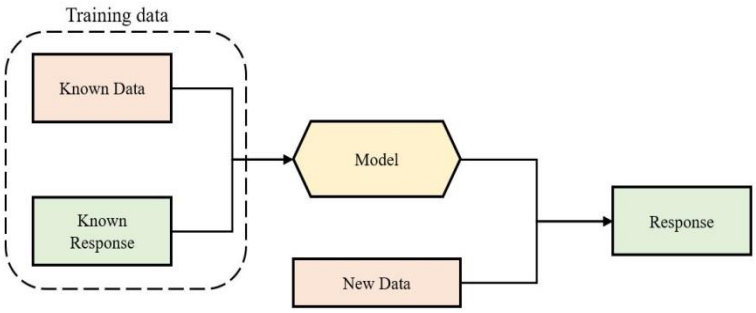


Figure 1.5: *Supervised learning steps*

2) Unsupervised Learning:

Unsupervised learning algorithms take a set of data that contains only inputs, and find structure in the data, like grouping or clustering of data points, without having previously an idea of what they can be or how many they should be, by figuring out the similarities between the given data with no human intervention.

3) Reinforcement Learning:

Reinforcement learning (RL) is called approximate dynamic programming, or neuro-dynamic programming, it is an area of machine learning concerned with how software agents ought to take actions in an environment in order to maximize the notion of cumulative reward.

Basic reinforcement is modeled as a Markov decision process [118]:

- A set of environment and agent states, S.
- A set of agent actions, A
- $P_a(s, s') = \Pr(s_{t+1} = s' | s_t = s, a_t = a)$ is the probability of transition (at time t) from state s to state s' under action a.
- $R_a(s, s')$ is the immediate reward after the transition from s to s' with action a.

4.2.3. Algorithms used in machine learning:

Type	Supervised Learning	Unsupervised Learning	Reinforcement Learning
Algorithms	Classification [162]. Regression [163].	Clustering [164]. Anomaly Detection [165]. Neural Networks [166]. Dimensionality reduction [167].	Criterion of optimality [168]. Brute force [169]. Value function [170]. Direct policy search [171].

Table 1.4: *Most common algorithms used in supervised, unsupervised and reinforcement learning.*

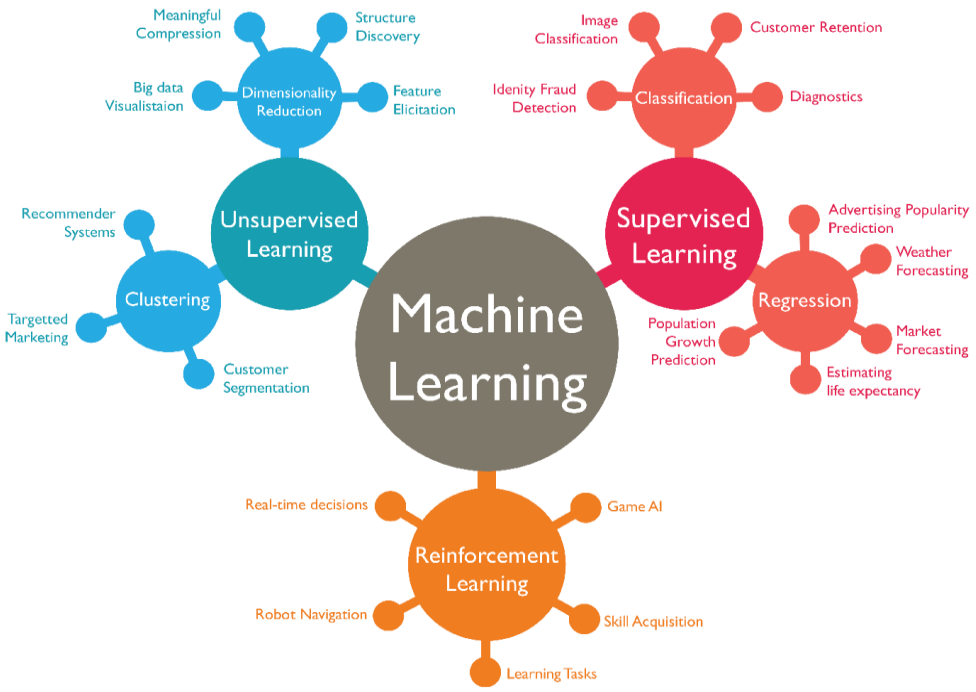


Figure 1.6: *Machine Learning Algorithms application field.*

4.2.4. Machine Learning Steps:

The result of the next steps is what's called a "model" [14], many successful models already exists in the field of Face Recognition (see ***table 1.7***).

1) Preparing the data and choose an algorithm

There are various learning algorithms (***figure 1.6***), each has its strengths and weaknesses, and no single one works best on all problems. In order to choose the best data presentation and most fitting algorithm for our problem, we must take in consideration the following tradeoffs:

- **Bias-variance tradeoff:**

The bias-variance dilemma is the conflict in trying to simultaneously minimize these two sources of error that prevent supervised learning algorithms from generalizing beyond their training set. high bias can cause an algorithm to miss the relevant relations between features and target outputs, where high variance can cause it to model the random noise in the training data rather than intended outputs (overfitting) [15]

- **Function complexity and amount of training data**

The amount of training data available relative to the complexity of the "true" function is an important tradeoff. If the true function is simple, then an "inflexible" learning algorithm with high bias and low variance will be able to learn it from a small amount of data. But if the true function is highly complex, means it behaves differently in different parts of the input

space, the function will only be able to learn from a very large amount of training data and using a "flexible" learning algorithm with low bias and high variance [16].

- **Dimensionality of the input space**

A third issue is the dimensionality of the input space. If the input feature vectors have very high dimensions, we stumble into the curse of dimensionality, which refers to all the problems that arise when working with data in the higher dimensions. The many "extra" dimensions can confuse the learning algorithm and cause it to have high variance means Overfitted. Hence, high input dimensional typically requires tuning the algorithm to have low variance and high bias.

In practice, human can intervene and manually remove irrelevant features from the input data, this is likely to improve the accuracy of the learned function. In addition, many algorithms for feature selection seek to identify the relevant features and discard the irrelevant ones. This is an instance of the more general strategy of dimensionality reduction, which seeks to map the input data into a lower-dimensional space prior to running the learning algorithm.

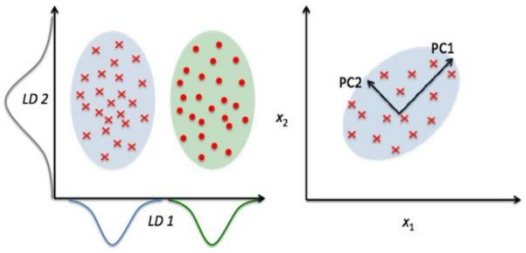


Figure 1.7: *LDA projects the data to signify the class separability, whereas PCA orients data along the direction of the component with maximum variance.*

The most common and well-known dimensionality reduction methods are the ones that apply linear transformations, PCA (Principal Component Analysis), Factor Analysis, and LDA (Linear Discriminant Analysis). (***figure 1.7***)

Whereas, the non-linear dimensionality reduction methods are MDS (Multi-dimensional scaling), Isomap (Isometric Feature Mapping), LLE (Locally Linear Embedding), HLLE (Hessian Eigenmapping), Laplacian Eigenmaps (Spectral Embedding), t-SNE (t-distributed Stochastic Neighbor Embedding. (***figure 1.8)***

To resume, the dimensionality reduction offers us many advantages such as:

- Less misleading data means model accuracy improves.
- Fewer dimensions mean less computing. Less data means that algorithms train faster.
- Less data means less storage space required.
- Fewer dimensions allow usage of algorithms unfit for a large number of dimensions

- Removes redundant features and noise**.**

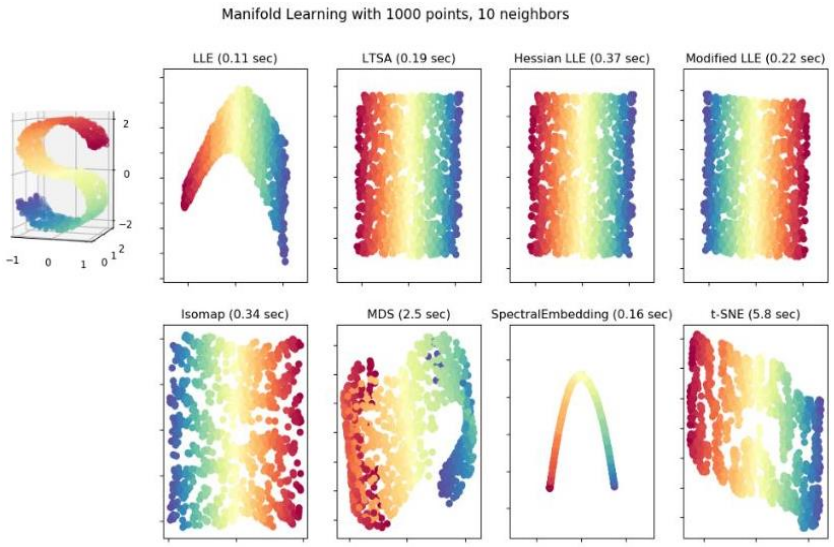


Figure 1.8: *The projection from applying different manifold learning methods on a 3D S-Curve*

2) Fitting/Training a model

When training for machine learning, an algorithm is passed with the training data. The learning algorithm finds patterns in the training data such that the input parameters correspond to the target. The output of the training process is a machine learning model which you can then use to make predictions. This process is also called "learning". Fitting or Training the model is applying the chosen algorithm on the chosen data with a human supervision using one of the validation methods mentioned next.

3) Choosing a validation method

There are different validation techniques used to examine the accuracy of the resulting fitted model:

- Resubstitution [172]
- Hold-out [173]
- K-fold cross-validation error [174]
- Bootstrapping [175]
- Random subsampling [175]
- LOOCV (Leave-One-Out Cross-Validation) [176]

4) Examining fit and update until satisfied

After validating the model, we might want to change it for better accuracy, better speed, or to use less memory as follows:

- Change fitting parameters to try to get a more accurate model.

- Change fitting parameters to try to get a smaller model. This sometimes improve accuracy.
- Try a different algorithm.

5) Using fitted model for predictions

This is the last step in machine learning, after being satisfied with the accuracy and details of the model, it is used to determine and classify a new unknown data.

4.3. Deep Learning:

Deep learning is the newest field in computer science, based on what's called Artificial Neural Networks, which is a technology inspired by the brain's neural network architecture, ANN are one of the most popular machine learning algorithms at present, it proved its capability at outperforming other algorithms in accuracy and speed. In this part, we present a fundamental understanding of what a neural network is starting from its most basic building block, which is a neuron, and later diving into its most popular types like CNN, RNN, etc.

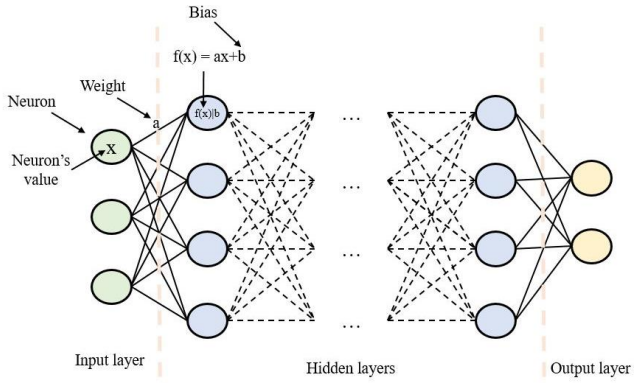


Figure 1.9: *Artificial Neural Network Architecture with its basic elements*

Deep learning is an AI function that mimics the work of the human brain in processing data for use in detecting and recognizing objects, speech, translating languages, and making decisions. It uses multiple layers to progressively extract higher-level features from the raw input. For example, in image processing, lower layers may identify edges, while higher layers may identify the concepts relevant to a human such as digits or letters or faces [23].

Deep learning is a subset of machine learning that is able to learn without human supervision, drawing from data that is both unstructured and unlabeled. Creating patterns for use in decision making. Deep learning unravels the huge amounts of unstructured data (Big Data) that would normally take humans decades to understand and process.

4.3.1. Deep Learning terminology:

1) Artificial Neuron

The functionality of the artificial neuron is similar to that of a human (***figure 1.10***), it takes in an input and returns an output, however, in mathematical terms, a neuron is a placeholder for a mathematical function, it applies the function given on the input and provides the result, which is the output. The functions used inside a neuron are generally termed as an Activation function. There are many types of these functions but there have been 5 major activation functions tried to date, step, sigmoid, tanh, ReLU and Leaky ReLU (***table 1.5***).

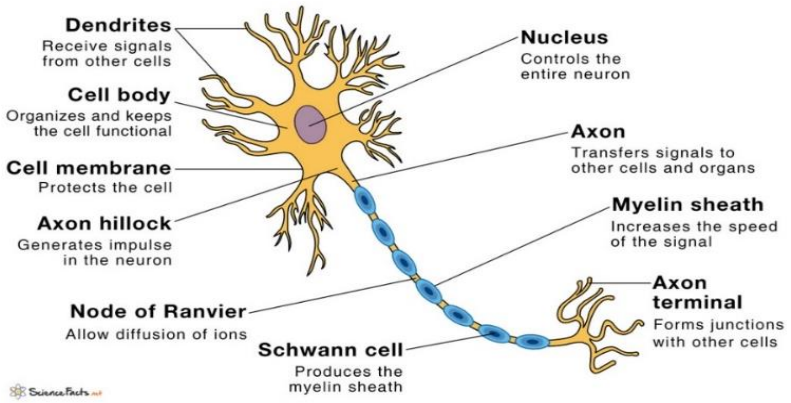


Figure 1.10: *Neuron's parts and their functions in Anatomy*

2) Layers

Artificial neural network architecture is defined by a collection of connected units or nodes (neurons); the ANN is combined of three main layers type, the input, the hidden and the output. Usually the number of layers is small and known, but in Deep Learning the layers number raised to thousands and some Deep Learning ANNs have unknown number of layers, hence why, it is called 'Deep'.

- **Input Layer**

There is one input layer in ANN, each type of ANN receives different kinds of data (information) depending on the aimed learning field, for instance, in Computer Vision, the first layer consists of neurons holding the pixel's intensities; for example, an input layer containing 1024 neurons, expects an image of size 32x32. The terminology layer and neuron here is just a way to help us humans perceive the data better, where in application, the image is simply flattened to an array of size 1024x1, the array is the layer and the array elements are the neurons. However in CNN, the image stays a matrix through the whole network and only flattened in the last what's called "fully-connected" layer.

- **Hidden layers**

Each network is unique, there is no specific number of layers, some networks use unknown number of layers between the input and output, hence it is called Hidden. The number of hidden layers depends on the data type, size, and the aimed learning, for example, for Face Detection in CNN, the network needs a layer to detect the edges (diagonal, vertical ..), a second to detect the combination of edges, which are shapes (circles, triangles, ..), a third to detect the face features (eyes, nose, ..), and another to detect the combination of features and decide whether it is a face or not.

- **Output Layer**

The output layer contains the results of the classification, for example, in Computer vision and hand-writing digits recognition, the output layer contains 10 neurons according to the number of digits used (0,1,2,..,9), after the many calculations in hidden layers, the last layer provide an estimation of what could the digit in the input image be. The decision is relying on mathematical function called the activation functions (***table 1.5***).

3) Weights

Neural networks connect neurons from layer to layer by weighted associations, the networks learn (or are trained) by processing examples containing a known "input" and "output" pairs, forming probability-weighted associations between the two, which are stored within the data structure of the network itself. The network then adjusts its weighted associations according to a learning rule and using the error value. Successive adjustments will cause the neural network to produce output, which is increasingly similar to the target output. After a sufficient number of these adjustments, the training can be terminated based upon certain criteria. This is known as supervised learning. And the result is called model [143].

4) Activation Functions

A given node (neuron) takes the weighted sum of its inputs, and passes it through a non-linear activation function. This is the output of the node, which then becomes the input of another node in the next layer; The activation function is what decides whether the neuron is fired (activated) or not, for example in Computer Vision and image processing, if two neurons detected two different edges that construct a corner, and one of the neurons in the next layer is specified for this corner, the activation function will ‘activate’ the neuron after receiving the two positive signals and evidently send a signal to the next layer mentioning there is a corner in that certain place [17].

Many activation functions have been used, each has its own formula, hence different results, although, the functions can be all used for the same purpose, the learning result varies, thus in order to choose the right function, tests need to be done and compared [18].

Name	Function and graph	Pros and cons
Step (biniary)	$f(x) = \begin{cases} 0 \; if \; x < 0 \\ 1 \; if \; x \geq 0 \end{cases}$	👍 Binary Classification 👎 Doesn't work in multi-label classification 👎 The derivative for the gradient calculation is always 0 so impossible to update weights
Linear	$f(x) = ax$	👍 Binary and multiclass classification 👍 Highly interpretable 👎 The derivative correspond to "a" so the update of weights and biaises during the backprogation will be constant. 👎 Not efficient if the gradient is always the same.
Sigmoid	$f(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{e^x+1}$	👎 The output of each neuron can saturate. 👎 The best sensitivity is around the central point (0, 0.5). 👎 The algorithm cannot learn during the saturation (it's the source of the vanishing gradient problem, corresponding to the absence of direction in the gradient).
Tanh	$f(x) = 2 * Sigmoid(2x) - 1$	👍 Range [-1,1] 👍 The gradient is stronger than sigmoid (derivatives are steeper) 👎 Like sigmoid, tanh also has a vanishing gradient problem
ReLU [162]	$f(x) = x^+ = \max(0, x)$	👍 Easy to implement and very fast 👍 True 0 value 👍 Optimization is easy when activation function is linear 👍 Most used in the neural networks ecosystem 👎 It can't be differentiable when x = 0. The gradient descent can't be computed for this point 👎 "dying ReLU problem" 👎 Not appropriate for RNN class algorithms (RNN, LSTM, GRU)
Leaky ReLU	$f(x) = \begin{cases} 0.01x \; if \; x < 0 \\ x \quad\quad if \; x \geq 0 \end{cases}$	👍 Correct the "dying ReLU problem" 👍 Same compartment of the ReLU activation function for the part y=x
Parametric ReLU	$f(x) = \begin{cases} ax \; if \; x < 0 \\ x \quad if \; x \geq 0 \end{cases}$	👍 Generalize the ReLU activation function 👍 Avoid the "dying ReLU problem" 👍 The parameter "a" is learned by the neural network

e-ReLU	$f(x) = \begin{cases} a(e^x - 1) & if\ x < 0 \\ x & if\ x \geq 0 \end{cases}$	👍 Becomes smooth slowly until its output equal to -α whereas RELU sharply smoothes. 👍 Strong alternative to ReLU. 👍 Unlike to ReLU, ELU can produce negative outputs. 👎 For x > 0, it can blow up the activation with the output range of [0, ∞].
ReLU-6	$f(x) = \begin{cases} \max(0, x) & if\ x < 6 \\ 6 & if\ x \geq 6 \end{cases}$	👍 Eliminates the possibility to blow up the activation with the output range of [0, ∞] when x>0 👍 Fix the step functions and basic ReLU cons
Softplus	$f(x) = \ln(1 + e^x)$	
Softsign	$f(x) = \frac{x}{1+\|x\|}$	
Softmax	$f(x) = \frac{e^x}{\sum_{j=1}^{K} e^{x_j}}$	👎 It is a probability distribution hence the output is different taking into account the sum of exponential.
Swish	$f(x) = \frac{x}{1 + e^{-x}}$ $= \text{x} * \text{sigmoid(x)}$	👎 Differentiable on each point compared to ReLU.

Table 1.5: *The activation functions with their pros and cons.*

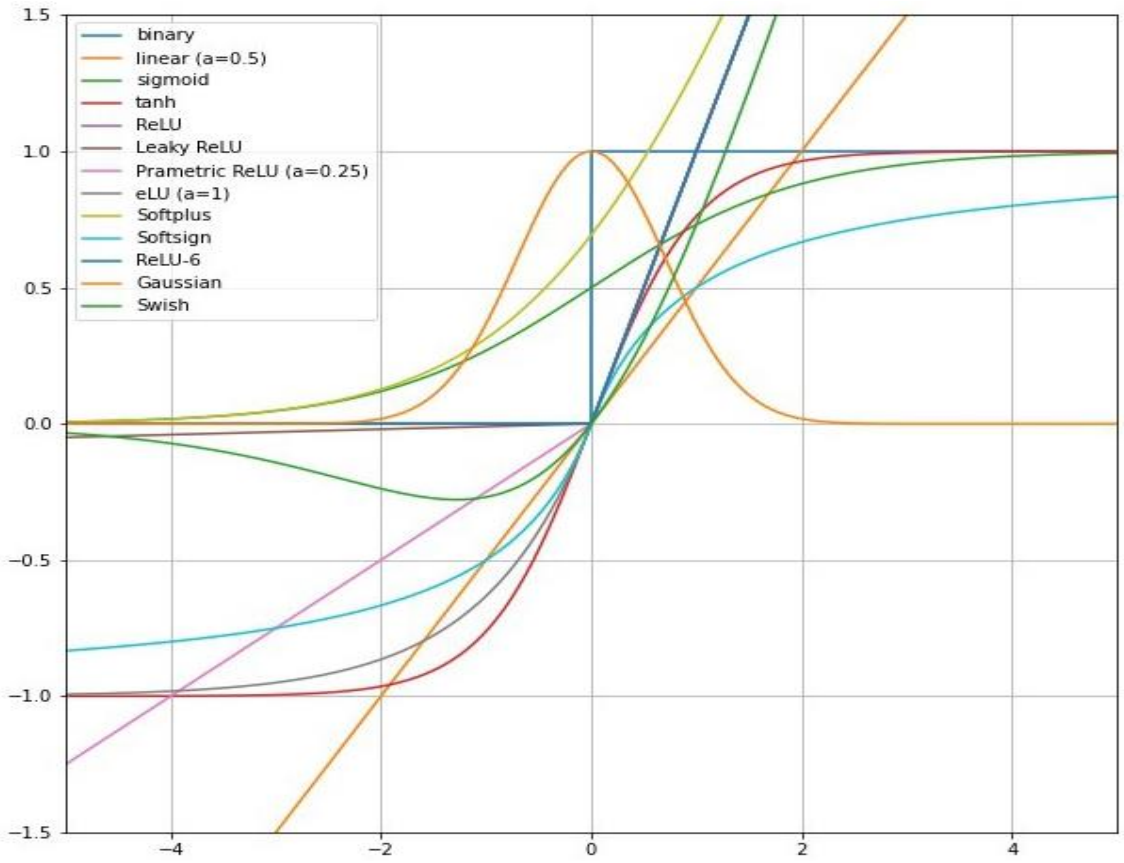


Figure 1.11: *Plot corresponds to the activation functions stack in one graphic.*

So we can conclure that the activation function defines the output of a neuron / node given an input or set of input (output of multiple neurons). It's the mimic of the stimulation of a biological neuron.

4.3.2. Artificial Neural Networks Types:

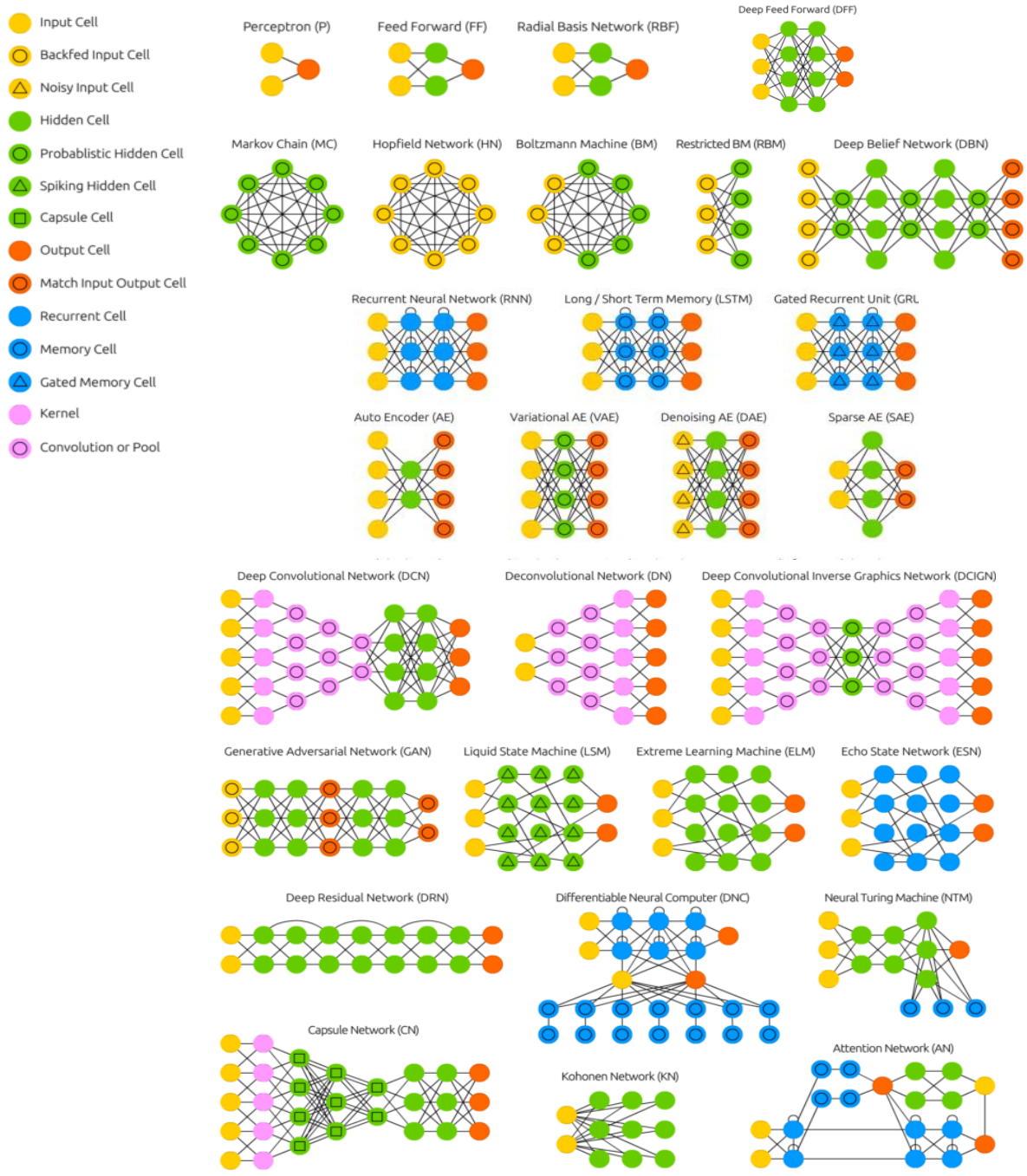


Figure 1.12: *A mostly complete ANN Chart [144].*

There are multiple Neural Networks types and architectures (***figure 1.12***), each have its own strengths and weaknesses, however, there are three mostly used ones, which are:

1) Convolutional Neural Network

CNN or Convolutional Neural Network is a neural network type that is heavily used in Computer Vision due to its capability of obliterating the need for fully connected layers during the execution. Regarding the size of the images used, the neurons in the first layer represent all the pixels in the image. Hence, it is necessary to find a neural network that can extract the features from the image and convert it into lower dimensions without losing its characteristics. [19] (View ***Chapter 2*** for more details on CNN*)*

2) Recurrent Neural Networks

A recurrent neural network (RNN) is a class of artificial neural networks in which node-to-node form a directed graph along a time continuum. This enables it to display temporal dynamic behavior. RNNs are derived from feedforward neural networks; thus, they can use their internal state (memory) to process variable length sequences of inputs. This makes them applicable to tasks such as unsegmented, connected handwriting recognition, or speech recognition [21].

RNNs are a sequential data processing family of neural networks. Much as a CNN which is specialized for processing a grid of values such as an image, an RNN is specialized for processing a sequence of values $x^{(1)}$, …, $x^{(\tau)}$, and just as CNN can readily scale to images with large width and height, and some CNN can process images of variable sizes, RNN can scale to much longer sequences than would be practical for networks without sequence-based specialization. Most recurrent networks can also process sequences of variable length [21].

3) LSTM

Long short-term memory (LSTM) is an artificial recurrent neural network (RNN) architecture, used in the field of deep learning. Unlike standard feedforward neural networks, LSTM has feedback connections. It can not only process single data points (such as images), but also entire sequences of data (such as speech or video) [22].

LSTM networks are well-suited to classifying, processing and making predictions based on time series data, since there can be lags of unknown duration between important events in a time series. LSTMs were developed to deal with the vanishing gradient problem that can be encountered when training traditional RNNs. Relative insensitivity to gap length is an advantage of LSTM over RNNs, hidden Markov models and other sequence learning methods in numerous applications [22].

4.3.3. Artificial Neural networks Application Fields

CNN	RNN	LSTM
▪ Image recognition ▪ Electromyography (EMG) recognition ▪ Video analysis ▪ Natural language processing ▪ Anomaly Detection ▪ Drug discovery ▪ Health risk assessment and biomarkers of aging discovery ▪ Checkers game ▪ Go the game ▪ Time series forecasting	▪ Machine Translation ▪ Robot control ▪ Time series prediction ▪ Speech recognition ▪ Speech synthesis ▪ Time series anomaly detection ▪ Rhythm learning ▪ Music composition ▪ Grammar learning ▪ Handwriting recognition ▪ Human action recognition ▪ Protein Homology Detection ▪ Predicting subcellular localization of proteins ▪ Several prediction tasks in the area of business process management. ▪ Prediction in medical care pathways	▪ Robot control ▪ Time series prediction ▪ Speech recognition ▪ Rhythm learning ▪ Music composition ▪ Grammar learning ▪ Handwriting recognition ▪ Human action recognition ▪ Sign language translation ▪ Protein homology detection ▪ Predicting subcellular localization of proteins ▪ Time series anomaly detection ▪ Several prediction tasks in the area of business process management ▪ Prediction in medical care pathways ▪ Semantic parsing ▪ Object co-segmentation ▪ Airport passenger management ▪ Short-term traffic forecast

Table 1.6: *The application fields of the three most used ANN type (CNN, RNN, LSTM).*

5. Deep extraction

The architectures can be classified into backbone networks and assembled networks, inspired by the extraordinary success of the ImageNet challenge, typical architectures from CNN, such as AlexNet, VGGNet, GoogleNet, ResNet and SENet, are presented and widely used as a basic model in FR (directly or slightly modified). In more than the mainstream, there are still some new architectures designed for FR to improve performance. In addition, when the basic networks are adopted as blocks basic FR methods often form assembled networks with inputs multiple or multiple tasks. A network is intended for one type of input or one type of task. Hu et al [28] show that it makes it possible to increase performance after accumulation of the results of the assembled networks.

6. Matching faces with deep features

Once the models are formed with big data and an appropriate loss function, each of the test images is passed through the networks to get an in-depth representation of the features. Once the characteristics deep extracted, most methods directly calculate the similarity between two characteristics using the cosine distance, then the nearest neighbor (PPV) and the Threshold comparison are used for identification and verification tasks [27].

7. Deep Learning models and architectures for FR

The training procedure to obtaining a functioning model is based on the following steps:

- Randomly initialize the weights for all the nodes. There are smart initialization methods.
- For every training example, perform a forward pass using the current weights, and calculate the output of each node going from left to right. The final output is the value of the last node.
- Compare the final output with the actual target in the training data, and measure the error using a loss function.
- Perform a backwards pass from right to left and propagate the error to every individual node using backpropagation [145]. Calculate each weight's contribution to the error, and adjust the weights accordingly using gradient descent. Propagate the error gradients back starting from the last layer.

Deep learning networks have established themselves as a promising model for face recognition. Their success is attributed towards multiple processing layers in order to learn data representations with several feature extraction levels. CNN have been presented as the deep learning tool in almost all face recognition systems. The significant breakthrough made by DeepIDs, DeepFace, Face++, FaceNet, and Baidu has changed the entire investigation scope. The deep face recognition techniques leverage hierarchical architecture in order to learn discriminative face representation. It has improved the system's performance appreciably which has led to the growth of several successful applications [29].

7.1. State of the art models in Deep FR

Model	Accuracy	Extra training	Paper	Year
VarGFaceNet	99.85%	✗	[118]	2019
ArcFace + MS1MV2 + R100	99.83%	✗	[119]	2018
PFEfuse+match	99.82%	✗	[120]	2019
VarGNet	99.733%	✗	[121]	2019
CosFace	99.73%	✗	[122]	2018
Dyna. AdaCos	99.73%	✗	[123]	2018
PAENet	99.67%	✓	[124]	2019
Seesaw-shuffleFaceNet (mobi)	99.65%	✗	[125]	2019
FaceNet	99.63%	✗	[126]	2015
DeepID3	99.53%	✗	[127]	2015
Ring loss	99.52%	✗	[128]	2018
DeepID2+	99.47%	✗	[129]	2014
SphereFace	99.42%	✗	[130]	2017
Light CNN-29	99.33%	✗	[131]	2015
Git Loss	99.30%	✗	[132]	2018
CPG	99.30%	✓	[133]	2019
Dynamic MTL	99.21%	✗	[134]	2019
DeepID2	99.15%	✗	[135]	2014
SeqFace	99.03%	✗	[136]	2018
SeetaFace	98.62%	✗	[137]	2016
GaussianFace	98.52%	✗	[138]	2014
VGG+GANFaces	94.9%	✗	[139]	2018
3DMM face shape parameters + CNN	92.35%	✗	[140]	2016

Table 1.7: *State-of-the-art models for face recognition (accuracy tested on LFW dataset)*

7.2. State of the art Deep Learning architectures in the field of Computer Vision:

2015	2016	2017	2018	2019	2020
Inception2 [37]	Inception4[45]	CapsNet[55]	YOLO v3[61]	YOLO v4[63]	CSPDarknet53[110]
Inception3 [44]	ResNets[46]	YOLO v2[56]	MobileNets2[62]	EfficientNet[88]	Deep-Captcha[111]
ResNet[38]	ResNext[47]	MobileNets[57]	MnasNet[76]	HRNet[89]	RegNetX[112]
SegNet[39]	ENet[48]	RefineDet[58]	ShuffleNet2[77]	MobileNets3[90]	RegNetY[112]
VGG[40]	SqueezeNet[49]	RetinaNet[59]	AmoebaNet[78]	Single-path NAS[91]	Assemble-ResNet[113]
Fast R-CNN [41]	YOLO v1[50]	DeformableCNN[60]	DetNet[79]	SNet[92]	TResNet[114]
Faster R-CNN[42]	Xception[51]	ShuffleNet[70]	PeleeNet[80]	MixNet[93]	ResNeSt[115]
Highway [43]	SSD[52]	SENet[71]	FBNet[81]	SpineNet[94]	GreedyNAS-A[116]
PReLU-Net [65]	DenseNet[53]	CheXNet[72]	SqueezeNext[82]	VoVNet[95]	GreedyNAS-B[116]
	DarkNet[66]	DPN[73]	ProxylessNet[83]	CSPResNeXt[96]	GreedyNAS-C[116]
	WideResNet[67]	RevNet[74]	DenseNet-Elastic[84]	VoVNet2[97]	Harm-Net[117]
	PyramidNet[68]	McKernel[75]	ResNet-D[85]	DenseNAS-C[98]	
	FractalNet[69]		ESPNet2[86]	ScaleNet[99]	
	SimpleNet[54]		Big-Little Net[87]	DiCENet[100]	
				MoGA-A[101]	
				ECA-Net[102]	
				GhostNet[103]	
				DenseNAS-A[104]	
				MultiGrain[105]	
				RandWire[106]	
				SKNet[107]	
				SCARLET[108]	
				DetNASNet[109]	

Table 1.8: *State-of-the-art DL architectures in the field of computer vision the past 5 years.*

Deep Learning Datasets for FR

The choice of the dataset is one of the two most important choices in deep learning after the architecture of the neural networks. Datasets are as the name suggests a set and a combination of data, it can be in different types depending on the field of use or the intentional learning. For example, in Computer Vision, the dataset is a collection of images (labeled in supervised learning, unlabeled in unsupervised learning). In Face Detection and Recognition for instance, the dataset is a set of face images in different variations to give the computer as many cases of the as possible in order for it to detect it and recognize it in any given situation.

The creation of a face dataset is based on so many criterions, from the image's visual parameters (lighting, color spaces, noises...), to the semantic part of the used images (Genders, Poses, Ages...).

Many face datasets already exists, each with its own specifics and details, some datasets focused on poses and others on ages. However, the newest technology has been 3D face reconstruction, that does not only take face recognition to another level, but gives the computer the ability to surpass human recognition systems.

7.3. State of the art of face datasets:

Name	Year	Images	classes	Type	Format	Resolution	Gender M/F	Age	Ethnicity	Landmarks	Poses	Illumination	Expressions	Occlusion	Size	Free
Yale [146]	1997	165	15	Gs	GIF	-	-	-	-	-	Frontal	Center Left Right	6 (Neutral Happy Sad Sleepy Surprised Wink)	Glasses	6.4 MB	✓
FERET Color [147]	1993 - 1996	14126	1199	RGB	TIFF	256× 384	-	-	-	-	-	-	-	-	8.5 GB	✓
PIE [148]	2000	40000	68	RGB	RAW PPM	640× 486	-	All	-	-	13	43	4	Glasses Beard &more	40 GB	✓
Multi PIE [149]	2000	750000	337	RGB	JPG PNG	Low - High	235/102	27yo	60% Eur-Am 35% Asian 3% Af-Am	39-68 manual	15	19	6 (Neutral Smile Surprise Squint Disgust Scream)	Glasses	308 GB	✗
Yale B+ [150]	2001	16128	28	Gs	PGM	680× 480	-	-	Very few	-	-	64	-	-	2GB	✓

Name	Year	Images	classes	Type	Format	Resolution	Gender M/F	Age	Ethnicity	Landmarks	Poses	Illumination	Expressions	Occlusion	Size	Free
CAS-PEAL-R1 [151]	2002 - 2003	30900	1040	Gs	BMP	320× 240	595/445	All	Asian	-	3 (Frontal Up Down)	15 (Ambiant Fluorescent Incandescent Elevation+ 45° Elebvation 0° Elevation- 45° And more)	6 (Neutral Smile Frown Surprise Eyes closed Mouth open)	Glasses Hats	26.6 GB	✓
FiA [152]	2004		180	RGB	MP4 (20s)	-	-	-	-	-	-	varies	-	-	257 GB	✗
CelebA [153]	2005	202599	10177	RGB	JPG	218× 178	84434/118165	-	Varies but Mostly White	5	Varied	Varied	Varied	Glasses Bangs Hats And more	-	-

Name	Year	Images	classes	Type	Format	Resolution	Gender M/F	Age	Ethnicity	Landmarks	Poses	Illumination	Expressions	Occlusion	Size	Free
LFW [154]	2007	13233	5749	RBG	JPEG	250× 250	Mostly men	10-80yo	Very few	-	-	Poor	-	Strong	232 MB	✓
SCface DB [155]	2011	4160	130	RGB	JPEG	Not fixed	115/15	20-75yo	Caucasians	21 manual	9	Varied	-	-	-	-
3DMAD [156]	2013	76500	17	RGB	MP4 (300 frame)	680× 480	-	-	-	-	-	-	-	-	58 GB	✓
FFHQ [157]	2019	70000	-	RGB	PNG	1024× 1024	-	-	Varied	-	Varied	Varied	Varied	Glasses Hats And more	2.56 TB	✓
AR [158]	1998	4000	126		JPG	576× 768	70/56	-			Frontal	Left on Right on Both on	Neutral Smile Anger Scream	Sunglasses Scarf	-	
CVL [159]	2003	798	114		JPEG	640× 480	Male (90%)	Young			Frontal Left/Right 45 Degrees left/right	Uniform	Serious Smile (teeth) with/without	None	-	

Name	Year	Images	classes	Type	Format	Resolution	Gender M/F	Age	Ethnicity	Landmarks	Poses	Illumination	Expressions	Occlusion	Size	Free
Muct [160]	2010	3755	276		-	640× 480	50% 50%	All			Frontal Left/Right Up/Down	Low Medium High	Neutral Smile	Glasses Headdresses	-	
UTKFace [161]	2017	20000	-	RGB	JPG	Not fixed	-	0-116yo	All	68	Varied	Varied	-	-	1.5 GB	✓
VGG Face 2 [218]	2018	3.31M	9131	RGB	JPG	Not fixed	M>F	All	different	-	Varied	Varied	Varied	Varied	37 GB	✓
CASIA web [219]	2014	494414	10575	RGB	BMP	640× 480	-	All	Varied	-	Varied	Varied	Varied	Glasses And more		✓
CelebMask-hq [220]	2020	+30000	19	RGB	JPG	512× 512	-	-	varied	-	Varied	Varied	Varied	Hair Hat Glasses And more		✓
FFDB [221]	2010	2845	5171	RGB	JPG	low	-	-	Varied	-	Varied	Varied	Varied	Hat Glasses And more	550 MB	✓

Name	Year	Images	classes	Type	Format	Resolution	Gender M/F	Age	Ethnicity	Landmarks	Poses	Illumination	Expressions	Occlusion	Size	Free
Wider face [222]	2016	32202	393703	RGB	JPG	low	-	-	Varied	-	Varied	Varied	Varied	Scarf Makeup And more	5 GB	✓
IMDb - Wiki [223]	2015	523051	20284	RGB		Not fixed	-	-	-	-	Varied	Varied	Varied	Hat And more	3 TB	✓

***Table 1.9**: State-of-the-art datasets in the facial recognition field.*

8. Deep Learning Challenges:

Although deep learning techniques are proving its best and has been solving various complicated applications with multiple layers and high level of abstraction. It is surely accepted that the accuracy, acuteness, receptiveness and precision of deep learning systems are almost equal or may sometimes surpass human experts. To feel the exhilaration of victory, in today's scenario, the technology has to accept many challenges. So, here is the list of challenges which deep learning has to overcome is:

- Deep learning algorithms have to continuously manage the input data.
- Algorithms need to ensure the transparency of the conclusion.
- Deep learning technology requires expensive high performance GPUs and storage equipment.
- It needs improved methods for big data analytics.
- Deep networks are called black box networks, because of their mysterious mechanism.
- The presence of hyper-parameters and its complex design.
- Suffer from local minima.
- Computationally intractable.
- Need a large amount of data.
- Expensive for the complex problems and computations.
- No strong theoretical foundation.
- Difficult to find the topology, training parameters for the deep learning.
- Deep learning provides new tools and infrastructures for the computation of the data and enables computers to learn objects and representations.

9. Conclusion:

In this Chapter, we have presented the basics of face detection and recognition, as well as the newest approach and trend used in recent years, which is Deep Learning, starting from Artificial intelligence and Machine Learning concepts, then we presented the different deep learning architectures and state of the art face recognition models that are based on CNN, after, we have mentioned challenges and obstacles of DL technology and made comparison between the most recent works that have been developed for facial recognition.

This study allowed us to see computer science in a whole different vision, The past decades of hard work and studies have led to this astonishing technology of Deep Learning which is unlike the usual technologies by all means, it gives the computer the ability to think and literally mimic human behavior and in some cases surpass it, with a very small human

supervision, thus, working with Deep Learning does not require the human intervention to specify or manually program the details in order to accomplish a certain task like face detection and recognition.

Although, the works and as mentioned, surpassed human's capabilities, this technology still needs some progress and development to get to the point where computers think and act on their own (Artificial Intelligence). In the next chapter we present CNN in depth and how it is used to perform a Facial Recognition task, also, we mention the frameworks available for the implementation of CNN, plus the challenges and obstacles and suggest some future work.

II. Convolutional Neural Network

1. Introduction

In Deep Learning, Convolutional neural network (CNN or ConvNet) is a type of feed-forward artificial neural network in which the connectivity pattern between neuron is inspired by the organization of the human visual cortex.

With the development of deep learning, face recognition technology based on CNN (Convolutional Neural Network) has become the main method adopted in the field of face recognition. In this Chapter we studied the basic principles of CNN:

- The most important methods build in CNN such as convolution and pooling.
- The different layers presented in CNN architecture, from input to fully connected and output layer with the definition of their details from data type to the functions applied on it.
- Techniques used in improving the CNN, such as Normalization, Optimization and Regularization, and how to avoid overfitting or underfitting.
- The frameworks and the predefined functions used to implement the CNN.

2. Why CNN for Computer Vision

The average number of neurons in the adult human primary visual cortex in each hemisphere has been estimated at around 140 million. The visual cortex has small regions of cells that are sensitive to specific regions of the visual field. Some individual neuronal cells in the brain reply (or fires) only in the presence of edges of a certain orientation. For example, some neurons fires when exposed to vertical edges and some when shown horizontal or diagonal edges [142].

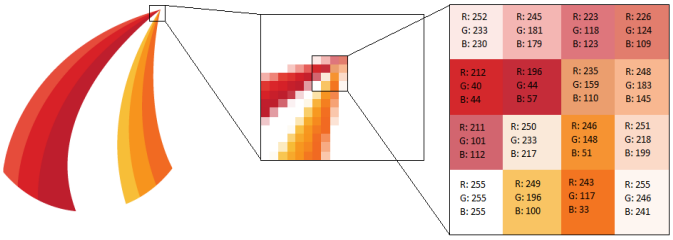


***Figure 2.1*:** How does the computer sees an image in RGB type

First, The computer sees the image as a collection of color intensities called a Pixel, in case of a colored RGB image, each pixel will have three values Red, Green, and Blue, in result, the image will be a matrix with three dimensions Width, Height, and Depth (depth=3 in RGB images).

Second, It is preferable to use convolutional neural networks for image classification rather than the traditional neural networks (multilayer perceptron MLP), in CNN the neuron in a layer will only be connected to a small region of the layer before it, unlike the neuron in MLP which is a fully connected networks, MLPs' cons are:

- MLP use one perceptron (neuron) for each input (e.g. pixel in an image, multiplied by 3 in RGB case). The amount of weights rapidly becomes unmanageable for large images. For a 224 x 224 pixel image with 3 color channels there are around 150,000 weights that must be trained! As a result, difficulties arise whilst training and overfitting can occur.
- MLPs react differently to an input (images) and its shifted version, they are not translation invariant. For example, if a picture of a face appears in the top left of the image in one picture and the bottom right of another picture, the MLP will try to correct itself and assume that a face will always appear in this section of the image (***figure 2.2***).

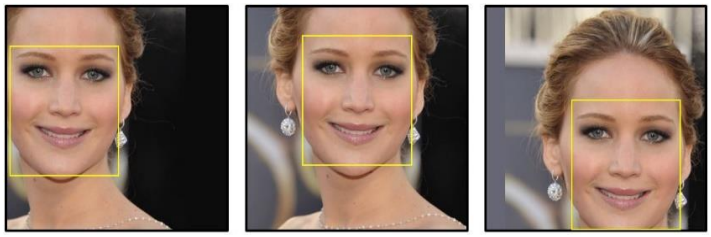


***Figure 2.2*:** Shifted version of the same image

3. CNN Architecture:

CNN's basic architecture includes these layers (***figure 2.3***), the main difference between the existing works that uses CNN as a base is the choice and the order of these layers and methods and its parameters.

Figure 2.3: *Basic CNN architecture*

3.1. Input Layer:

Input layers in CNN hold the raw pixel values of the image. Image data is represented by 3D matrix (width × height × depth). In traditional MLP, the image needs to be reshaped into a single column. Supposing an image of dimension 28 x 28 =784, it needs to be converted into 784 x 1 before feeding it to the input layer. If there are "m" training examples then dimension of input will be (784, m). However, in CNN, the image remains of a shape Matrix.

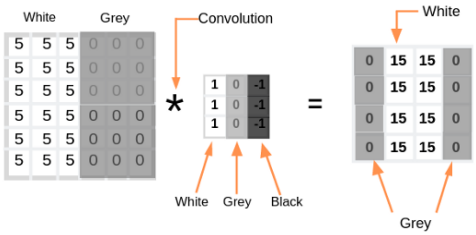


Figure 2.4: *Example of Convolution operation*

3.2. Convolution Operation:

Convolutional layer is sometimes called feature extractor layer, because features of the image are extracted within this layer. First of all, a part of the image is connected to convolution layer to perform convolution operation and calculating the dot product between receptive field (it is a local region of the input image that has the same size as that of filter) and the filter. Result of the operation is single integer of the output volume. Then the filter slides over the next receptive field of the same input image by a Stride and perform the same operation again. This process is repeated again and again until the image's end. The output will be the input for the next layer.

The convolution operation can be visualized in (***Figure 2.5)***. Here the image dimension is 4x4 and filter is 3x3, hence we are getting output after convolution is 2x2.

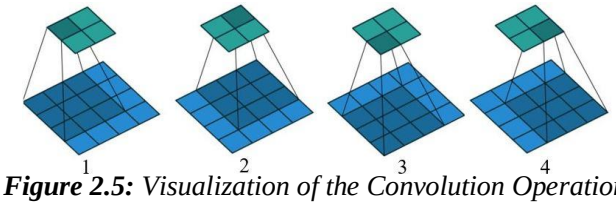


Figure 2.5: *Visualization of the Convolution Operation*

- **Filters:**

The filters applied in convolution are sets of cube-shaped weights that are applied throughout the image. Each 2D slice of the filters are called kernels. These filters introduce translation invariance and parameters sharing.

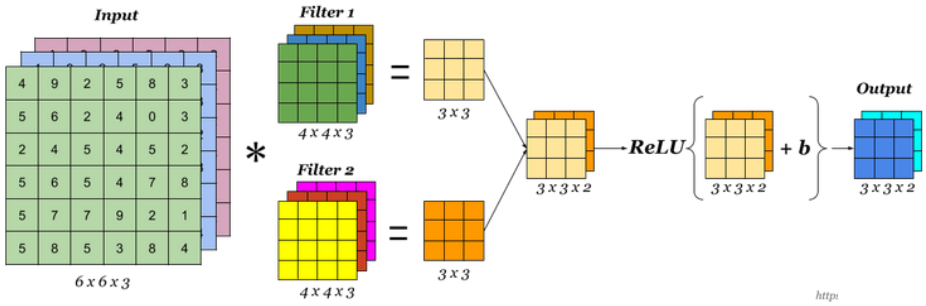
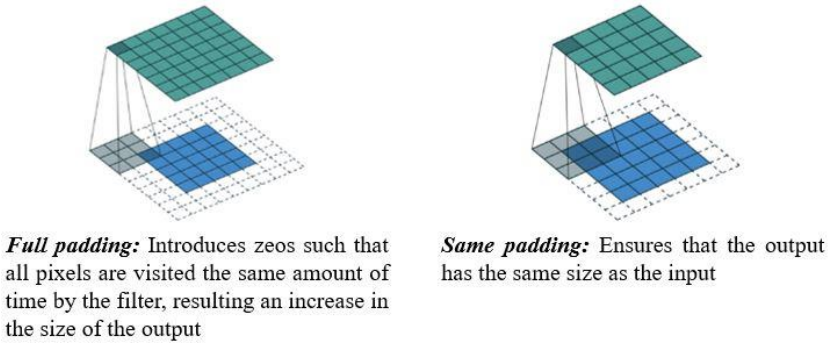


Figure 2.6: *Visualization of the use of filters in Convolution Operation*

- **Padding**

Applying convolutions on a normal image reside down the image by an amount depending on the size of the filter. Hence why padding is used to eliminate the loss of what can be an important part of the image.



Full padding: Introduces zeos such that all pixels are visited the same amount of time by the filter, resulting an increase in the size of the output

Same padding: Ensures that the output has the same size as the input

Figure 2.7: *Visualization of different paddings and their resulted output*

3.3. ReLU function:

Activation layers in CNN usually consists of the ReLU function, which is the most successful non-linear function; an element wise activation function, such as the max(0,x), thresholding at zero. Applying it leaves the size of the volume unchanged (see ***table 1.5***).

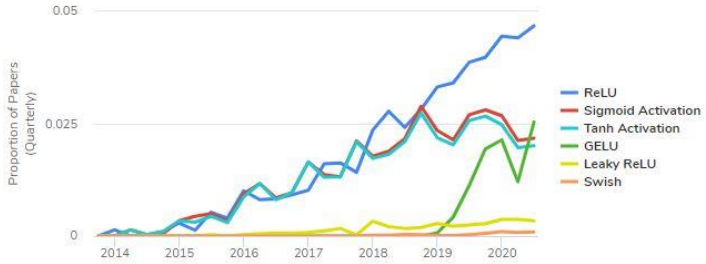


Figure 2.8: *Usage Over Time of Most known activation functions*

3.4. Pooling Operation:

Pooling is used to reduce the spatial volume of input image after convolutions. It is used between two convolutions. If FC (fully connected) is applied after Convo without applying pooling or max pooling, then it will be computationally expensive. The max pooling is the only way to reduce the spatial volume of input image. In the example bellow, max pooling is applied in single depth slice with Stride of 2. the 4x4 dimension input is reduced to 2x2 dimensions.

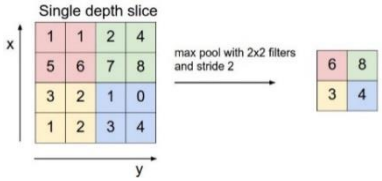


Figure 2.9: *Example of Max Pooling in CNN*

There is no parameter in pooling layer but it has two hyper-parameters — Filter (F) and Stride(S). In general, if we have input dimension W1 x H1 x D1, then the W2, H2, D2 of the output is:

$$W2 = (W1-F)/S+1,\ H2 = (H1-F)/S+1,\ D2 = D1$$

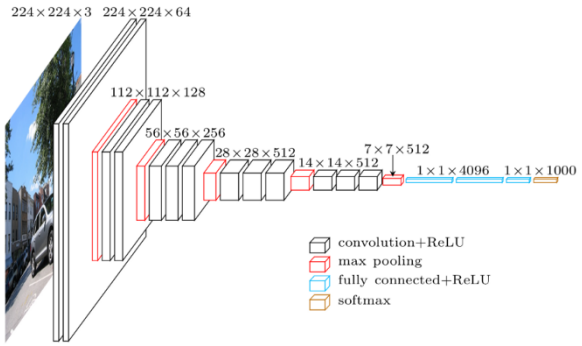


Figure 2.10: *Example of pooling (down sampling) in CNN*

- **Pooling types:**

Type:	**Description**	**Pros**
Max [176]	Calculates the maximum value for patches of a feature map, It adds a small amount of translation invariance.	Best at extracting more pronounced features like edges
Average [177]	Calculates the average value for patches of a feature map	Extracts features more smoothly than max pooling
Global [177]	With global pooling reduces the dimensionality from 3D to 1D. Therefore global pooling outputs 1 response for every feature map. This can be the maximum or the average or whatever other pooling operation used. It is often used at the end of the backend of a convolutional neural network to get a shape that works with dense layers.	Reduces the dimensionality from 3D to 1D and eliminate the need to apply flattening
Spatial Pyramid [178]	SPP works by dividing the feature maps output by the last convolutional layer into a number of spatial bins with sizes proportional to the image size, so the number of bins is fixed regardless of the image size. Bins are captured at different levels of granularity	Maps any size input down to a fixed size output.

Cascade Corner [179]	It is a technique for object detection that seeks to better localize corners by encoding explicit prior knowledge. Suppose we want to determine if a pixel at location is a top-left corner. Let and be the feature maps that are the inputs to the top-left corner pooling layer, and let and be the vectors at location in and respectively. With feature maps, the corner pooling layer first max-pools all feature vectors between and into a feature vector, and max-pools all feature vectors between and into a feature vector. Finally, it adds and together.	Better localization of the corners by encoding explicit prior knowledge
Center [180]	The backbone outputs a feature map, and to determine if a pixel in the feature map is a center keypoint, we need to find the maximum value in its both horizontal and vertical directions and add them together. By doing this, center pooling helps the better detection of center keypoints.	Capture richer and more recognizable visual patterns. The geometric centers of objects do not necessarily convey very recognizable visual patterns.

Table 2.1: *Pooling types with their definitions and advantages*

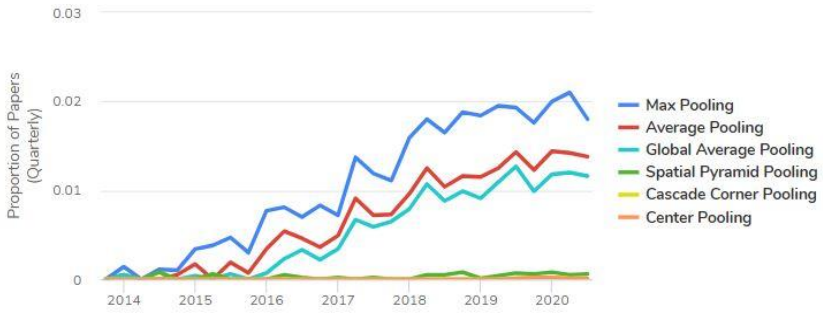


Figure 2.11: *Usage Over Time of Most known Pooling methods*

3.5. Fully Connected Layer:

Fully Connected Layer is a feed forward neural network. Fully Connected Layers form the last few layers in the network. The input to the fully connected layer is the output from the final Pooling or Convolutional Layer, which is flattened and then fed into the fully connected layer.

FC layer computes the class scores, as with ordinary Neural Networks and as the name implies, each neuron in this layer will be connected to all the neurons in the previous layer. As for the number of FC, it varies depending on the data used. Some famous CNN structure in ILSVRC, such as AlexNet, VGG, ZF net, etc. use two fully connected layer, followed by the output layer.

3.6. Softmax/Logistic function:

Softmax or Logistic function exists in the last layer of CNN. It resides at the end of FC layer. Logistic is used for binary classification and softmax is for multi-classification. It is a

form of multinomial logistic regression that normalizes an input value into a vector of values that follows a probability distribution whose total sums up to 1.

The output values are between the range [0,1] which enable us to avoid binary classification and accommodate as many classes or dimensions in our neural network model. As an aside, another name for Softmax Regression is Maximum Entropy (MaxEnt) Classifier [162].

The function is usually used to compute losses that can be expected when training a data set. Known use-cases of softmax regression are in discriminative models such as Cross-Entropy and Noise Contrastive Estimation. These are only two among various techniques that attempt to optimize the current training set to increase the likelihood of predicting the correct word or sentence.

3.7. Output Layer

The output layer in a CNN as mentioned previously is a fully connected layer, where the input from the other layers is flattened and sent so as the transform the output into the number of classes as desired by the network. The Output layer contains the label which is in the form of one-hot encoded.

The output of the CNN is also a 4D array. Where batch size would be the same as input batch size but the other 3 dimensions of the image might change depending upon the values of the filter, kernel size, and padding we use

3.8. Normalization

Normalization is an approach which is applied during the preparation of data in order to change the values of numeric columns in a dataset to use a common scale when the features in the data have different ranges. Hence, it is used to make optimization easier by smoothing the loss surface of the network. It normalizes each input channel across a mini-batch. To speed up training of convolutional neural networks and reduce the sensitivity to network initialization, it is used between convolutional layers and nonlinearities, such as ReLU layers [163].

- **Normalization Methods:**

#	Name	Reference	Year
1	Layer Normalization	Ba, J.L et al [164]	2016
2	Batch Normalization	Ioffe, S et al [165]	2015
3	Local Response Normalization	Krizhevsky et al [166]	2012

4	Instance Normalization	Ulyanov D et al [167]	2016
5	Spectral Normalization	Miyato T et al [168]	2018
6	Adaptive Instance Normalization	Ogasawara E et al [169]	2017
7	Weight Normalization	Salimans T et al [170]	2016
8	Conditional Batch Normalization	De Vries H et al [171]	2017
9	Group Normalization	Wu T et al [172]	2018
10	Activation Normalization	Kingma D.P et al [173]	2018
11	Weight Demodulation	Karras T et al [174]	2019
12	Switchable Normalization	Luo P et al [175]	2018
13	Local Contrast Normalization	Foracchia M et al [176]	2009
14	Weight Standardization	Qiao et al [177]	2019
15	Conditional Instance Normalization	Huang X et al [178]	2016
16	SyncBN	He K et al [179]	2018
17	Attentive Normalization	Li X et al [180]	2019
18	Decorrelated Batch Normalization	Huang L et al [181]	2018

Table 2.2:*Most used Normalization Methods ordered by their number of use*

3.9. Regularization

In mathematics, statistics, finance, computer science, particularly in machine learning and inverse problems, regularization is the process of adding information in order to solve an ill-posed problem or to prevent overfitting. Regularization applies to objective functions in ill-posed optimization problems.

Regularization strategies are designed to reduce the test error of a machine learning algorithms, possibly at the expense of training error. Many different forms of regularization exist in the field of deep learning

- **Dropout**

Dropout is the most used regularization technique, it is a technique where randomly selected neurons are ignored during training. They are “dropped-out” randomly. It is a stochastic regularization technique that reduces overfitting by (theoretically) combining many different neural network architectures. With Dropout, the training process essentially drops out neurons in a neural network. They are temporarily removed from the network, which can be visualized in (***figure 1.12).***

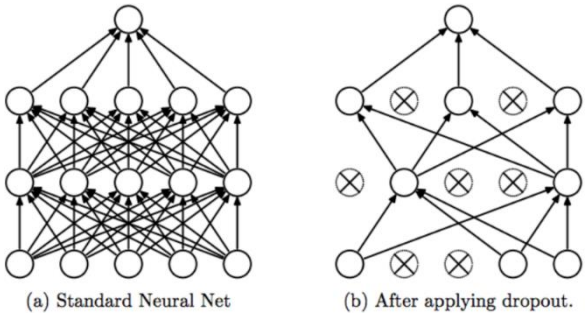


Figure 2.12: *Standard Neural Net VS After applying Dropout*

This removal of neurons and synapses during training is performed at random, with a parameter p that is tunable (or, given empirical tests, best set to 0.5 for hidden layers and close to 1.0 for the input layer). This effectively means that, according to the authors, the "thinned" network is sampled from the global architecture, and used for training.

It drops a unit (along with connections) at training time with a specified probability ***p*** (a common value is ***p= 0.5***). all units are present, but with weights scaled by ***p*** (i.e. becomes ***pw***). The idea is to prevent co-adaptation, where the neural network becomes too reliant on particular connections, as this could be symptomatic of overfitting. Intuitively, dropout can be thought of as creating an implicit ensemble of neural networks.

- **Regularization Methods:**

#	Reference	Name
1	Srivastava N et al [182]	Dropout
2	Loshchilov I et al [183]	Weight Decay
3	Choe J et al [184]	Attention Dropout
4	Pereyra G et al [185]	Label Smoothing
5	Grandvalet Y et al [186]	Entropy Regularization
6	Yao Y et al [187]	Early Stopping
7	Kingma D.P et al [188]	Variational Dropout
8	Wan L et al [189]	DropConnect
9	Elden L et al [190]	R1 Regularization
10	Park M.Y et al [191]	L1 Regularization
11	Gal Y et al [192]	Embedding Dropout
12	Kim S.J et al [193]	Off-Diagonal Orthogonal
13	Oster H.S et al [194]	Temporal Activation Regularization

Table 2.3: *Regularization Methods ordered by their number of use*

3.10. Optimization

Optimizers are algorithms or methods used to change the attributes of neural network such as weights and learning rate in order to reduce the losses. In the simplest case, an optimization problem consists of maximizing or minimizing a real function by systematically choosing input values from within an allowed set and computing the value of the function. The generalization of optimization theory and techniques to other formulations constitutes a large area of applied mathematics. More generally, optimization includes finding "best available" values of some objective function given a defined domain (or input), including a variety of different types of objective functions and different types of domains.

- **Most known Optimizers:**

Name	Advantages	Disadvantages
Gradient Descent	• Easy computation • Easy to implement • Easy to understand	• May trap at local minima • Weights are changed after calculating gradients on the whole dataset. So, if the dataset is too large, than this may take years to converge to the minima. • Requires large memory to calculate gradients on the whole dataset.
Stochastic Gradient Descent	• Frequent updates of model parameters hence, converges in less time. • Requires less memory as no need to store values of loss functions. • May get new minima's.	• High variance in model parameters. • May shoot even after achieving global minima. • To get the same convergence as gradient descent needs to slowly reduce the value of learning rate.
Mini-Batch Gradient Descent	• Frequently updates the model parameters and also has less variance. • Requires medium amount of memory.	
Momentum	• Reduces the oscillations and high variance of the parameters. • Converges faster than gradient descent.	• One more hyper-parameter is added which needs to be selected manually and accurately.
Nesterov Accelerated Gradient	• Does not miss the local minima. • Slows if minima's are occurring.	• Still, the hyperparameter needs to be selected manually.
Adagrad	• Learning rate changes for each training parameter.	• Computationally expensive as a need to calculate the second order derivative.

	• Don't need to manually tune the learning rate. • Able to train on sparse data.	• The learning rate is always decreasing results in slow training.
AdaDelta	• Now the learning rate does not decay and the training does not stop.	• Computationally expensive.
Adam	• The method is too fast and converges rapidly. • Rectifies vanishing learning rate, high variance.	• Computationally costly.

Table 2.4:*Advantages and Disadvantages of popular Optimization algorithms*

- ✓ Adam is the best optimizers, if we want to train the neural network in less time and more efficiently.
- ✓ For sparse data use the optimizers with dynamic learning rate.
- ✓ If, we want to use gradient descent algorithm than min-batch gradient descent is the best option.

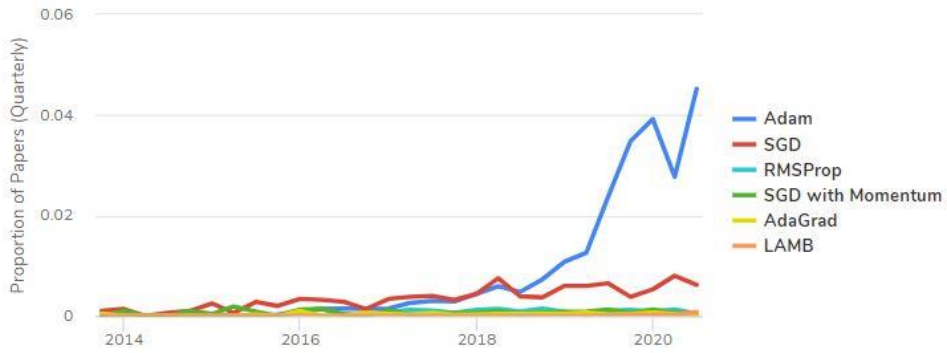


Figure 2.13: *Usage Over time of the most popular Optimization methods*

- **Most used Optimization methods:**

#	Name	Reference	Year
1	Adam	Kingma D.P et al [195]	2014
2	SGD	Robbins H and Monro S [196]	1951
3	ADMM	Boyd S et al [197]	2000
4	RMSProp	Hinton G [198]	2013
5	SGD with Momentum	Ning Q et al [199]	1999
6	AdaGrad	Duchi J et al [200]	2011
7	TTUR	Heusel M et al [201]	2017
8	Gradient Clipping	Pascanu et al [202]	2000

9	LAMB	You Y et al [203]	2019
10	AMSGrad	Reddi et al [204]	2019
11	Nesterov Accelerated Gradient	Nesterov Y [205]	1983
12	Adafactor	Shazeer M et al [206]	2018
14	Natural Gradient Descent	Rattray et al [207]	1998
15	LARS	You Y et al [208]	2017
16	Population Based Training	Jaderberg M et al [209]	2017

Table 2.5:*Most used Optimization Methods and ordered by their use rate*

4. Image Augmentation:

Image augmentation is a data augmentation method that generates more training data from the existing training samples. The common case in most machine learning applications, especially in image classification tasks, is that new training data is hard to obtain or expensive.

Data augmentation is a way to generate more training data from our current set. It enriches or "augments" the training data by generating new examples via random transformation of existing ones. This way we artificially boost the size of the training set, reducing overfitting (***figure 2.14***).

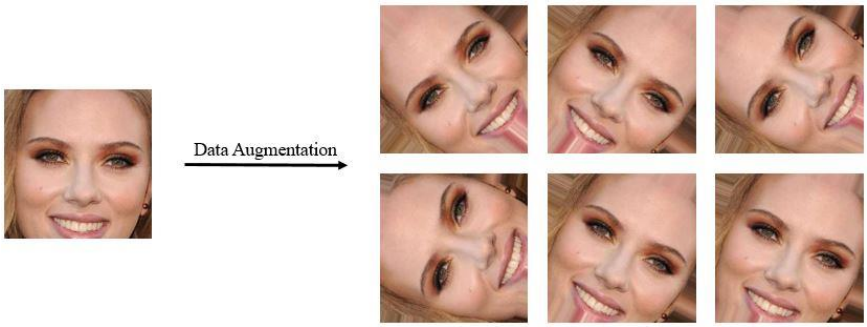


Figure 2.14: *Example on Data Augmentation in Computer Vision.*

Data augmentation is done dynamically during training time. Common transformations are: rotation, shifting, resizing, exposure adjustment, contrast change etc. data augmentation is only performed on the training data. The validation or test set remains unchanged.

5. Overfitting and Underfitting in CNN

Overfitting happens when the neural network is good at learning its training set, but is not able to generalize its predictions to additional, unseen examples. This is characterized by low bias and high variance. Underfitting happens when the neural network is not able to accurately predict for the training set, not to mention for the validation set. This is characterized by high bias and high variance (details in Chapter 1 -Machine learning-).

Methods to avoid Overfitting	Methods to avoid Underfitting
• **Retraining neural networks:** running the same model on the same training set but with different initial weights, and selecting the network with the best performance. • **Multiple neural networks:** training several neural network models in parallel, with the same structure but different weights, and averaging their outputs. • **Early stopping:** training the network, and monitoring the error on the validation set after each iteration, and stopping training when the network starts to over-fit the data. • **Regularization:** adding a term to the error function equation, intended to decrease the weights and biases, smooth outputs and make the network less likely to over-fit. • **Tuning performance ratio:** similar to regularization, but using a parameter that defines by how much the network should be regularized.	• **Adding neuron layers or inputs:** adding neuron layers, or increasing the number of inputs and neurons in each layer, can generate more complex predictions and improve the fit of the model. • **Adding more training samples or improving quality:** the more training samples you feed into the network, and the better they represent the variance in the real population, the better the network will perform. • **Dropout:** randomly "kill" a certain percentage of neurons in every training iteration. This ensures some information learned is randomly removed, reducing the risk of overfitting. • **Decreasing regularization parameter:** regularization can be overdone. By using a regularization performance parameter, you can learn the optimal degree of regularization, which can help the model to better fit the data.

Table 2.6:*Methods used to avoid both Underfiting and overfitting*

6. Frameworks used to implement CNN

Given that deep learning is the key to performing tasks of a higher level of sophistication, building them successfully is a proven to be challenging for data scientists and data engineers across the globe. Today, we have a big collection of frameworks at our disposal that allows us to develop tools that can offer a better level of abstraction along with simplification of difficult programming challenges.

Each framework is built in a different manner for different purposes. The top eight deep learning frameworks with their main highlights are:

Framework	Highlights
TensorFlow [210]	- Robust multiple GPU support. - Graph visualization and queues using TensorBoard. - Known to be complex and has a steep learning curve. - Excellent documentation and community support.

PyTorch [211]	- Excellent at rapid prototyping. - Strong support for GPUs as parallel programs can be implemented on multiple GPUs. - Provides cleaner interface and is easier to use. - Facilitates the exchange of data with external libraries.
DeepLearningG4J [212]	- Brings together the entire Java ecosystem to execute deep learning. - Can process massive amounts of data quickly. - Includes both multi-threaded and single-threaded deep learning frameworks. - Can be administered on top of Hadoop and Spark.
CNTK - The Microsoft Cognitive Toolkit [213]	- Highly efficient and scalable for multiple machines. - Supported by interfaces such as Python, C++, and Command Line. - Fit for image, handwriting and speech recognition use cases. - Supports both RNN and CNN type of neural networks.
Keras [214]	- Easy-to-understand and consistent APIs. - Seamlessly integrates with TensorFlow workflow. - Supports multiple deep learning backends. - Built-in support for distributed training and multi-GPU parallelism.
ONNX [215]	- Provides interoperability and flexibility. - Provides compatible runtimes and libraries. - Liberty of using the preferred framework with a selected inference engine. - Maximizes performance across hardware.
MXNet [216]	- Hybrid programming which provides the best of both imperative and symbolic programming. - Provides distributed training. - Supports deployment in different languages such as Java, Scala, R, Julia, C++, Perl, and Clojure. - Nearly linear on GPU clusters which provides excellent scalability.
Caffe [217]	- C++ library comes with a Python interface. - The configuration defines models without hard-coding. - Easier to set up and train, without having to build onto the network. - Support for recurrent neural networks is quite poor.

Table 2.7: *Top Eight Framework used in Deep Learning with their main highlights*

7. Conclusion

In this Chapter, we have presented the basics of CNN, starting from its main fundamentals such as Convolutions and Pooling, moving on to its layers types and their detailed functioning. then we presented the different ways to improve the work of the CNN using Normalization, Regularization, and Optimization methods. Next, the implementation of CNN and how to use it for computer vision in general. After, we have made comparison between the

most used frameworks to achieve the best recognition rate taking in consideration the storage space and the time available.

We learned that in other machine learning algorithms, the pictures need us to perform preprocessing or feature extraction. However, we rarely need to do these operations when using CNN for image processing. In terms of algorithms, there are sharing parameters between the convolution layers of CNN. The advantage of this is that the memory requirements are reduced, and the number of parameters to be trained is correspondingly reduced. The performance of the algorithm is therefore improved.

Today, very good results have been achieved in the field of face recognition and other computer vision applications, which will be framed in the next Chapter, where we dived into conception, code programming and implementation, and evaluated a CNN on our own face dataset called “Actors dataset”, where we obtained some impressive results and accuracy.

III. Conception, Implementation and Results

1. Introduction

After presenting the fundamentals of Facial Recognition systems and Artificial intelligence in chapter one, and taking deeper look into Convolutional neural network's details in chapter two, in this chapter we made our conception and implemented the method we have chosen to resolve the facial recognition problem using the newest state of the art Deep Learning technology.

Using Python programming language in Google Colab's free programming environment. This chapter focuses on the conception and details it into four main stages:

Stage 1: Loading, augmenting and pre-processing the data.

Stage 2: Defining, fine-tuning the model's architecture.

Stage 3: Training and fitting the model.

Stage 4: Estimating and testing the model's performance.

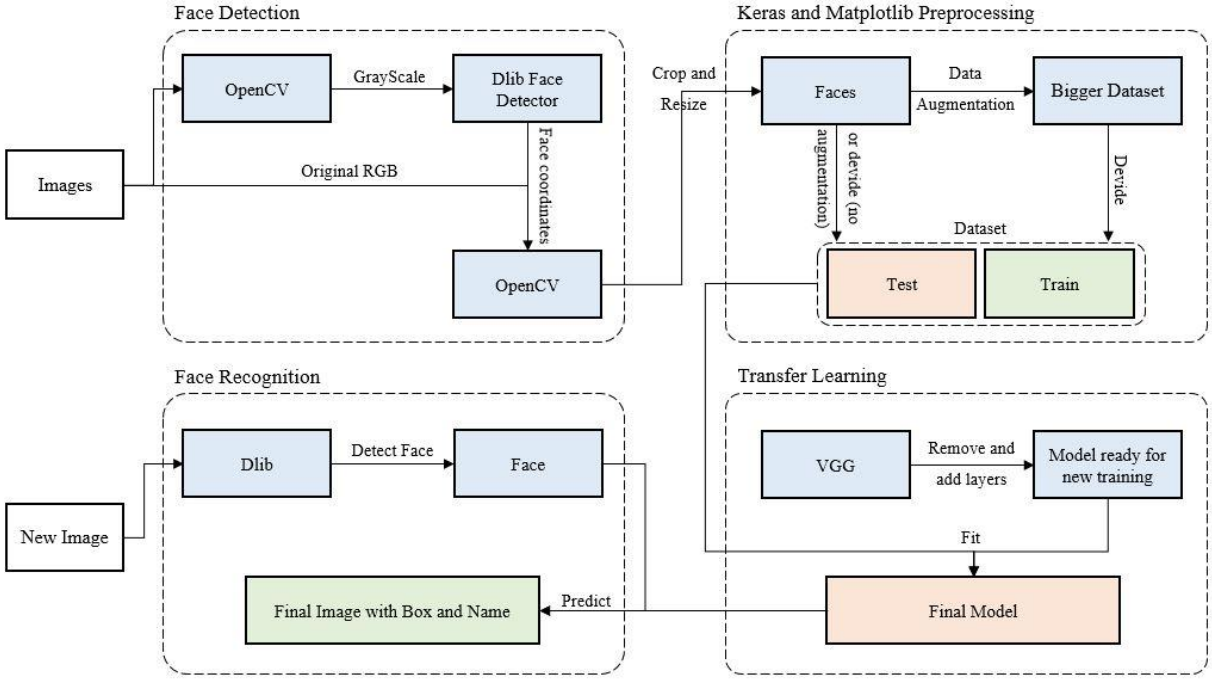


Figure 3.1: *Conception Graph*

2. Loading and pre-processing the data

2.1. Loading the Dataset:

Preprocessing data step consist of the dataset and its partitioning choice, the different mathematical image enhancement methods such as noise reduction, data augmentation…

We made a new dataset in this work called “Actors”, it has 12 classes (12 different known actors), with an average of 30 images per class, collected from the internet which becomes over 3600 images after applying data augmentation, the images are of JPG format with different resolutions and varies from high to low quality, with lightning variations and

Figure 3.2: *Example of our Actors Dataset*

background occlusion variation.

The samples of faces we have chosen are in different varieties; first: two facial expressions only (neutral and smiling), second: different facial occlusion (Glasses, moustache, beard, hat, makeup), the poses are mainly frontal with just few side variations, and there is age variation as well.

Here are the steps we did in order to load our data:

1) Connect Google Drive with Google Colab to easily manage the images and files.
2) Imported and used OpenCv to read the images and other python dependencies to manage the files in python.

2.2. Preprocessing the dataset:

The preprocessing consist of detecting, aligning, and cropping the faces, and these are steps:

1) Imported Dlib’s face detector which uses CNN.
2) Read and converted each image to grayscale using OpenCV.
3) Cropped the faces from the original RGB images using the coordinates from Dlib detector.

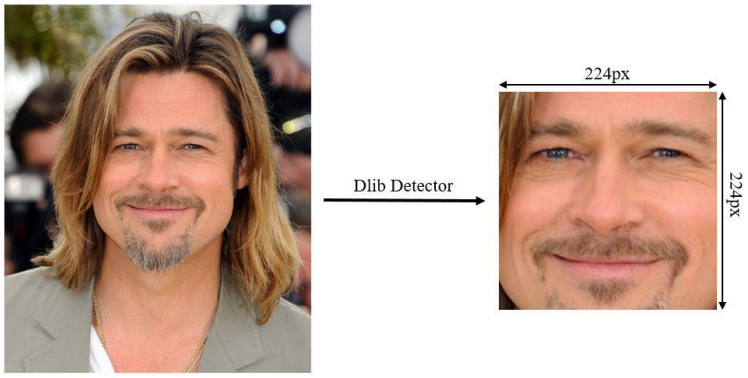


Figure 3.3: *Example Dlib face detector result*

4) Applied face augmentation, on the cropped images that resulted 9 different and new images for each image.
5) Divided the dataset to 80% learning set and 20% validating set.

3. Defining the model's architecture

Defining the model's architecture requires choosing the hyper-parameters of the model, which are:

- Number of Convolutional layers.
- Type of activation functions for each layer.
- Number of hidden units for each layer.

The choice of these hyper-parameters can be done by copying the existing research/studies and doing transfer learning on our own dataset. Or experimenting with new values until the best match is found, but this is a time consuming process. In our work we used the benchmark CNN architecture VGG16 model, which is combined of

- 16 convolutional layers.
- 5 max pooling layers

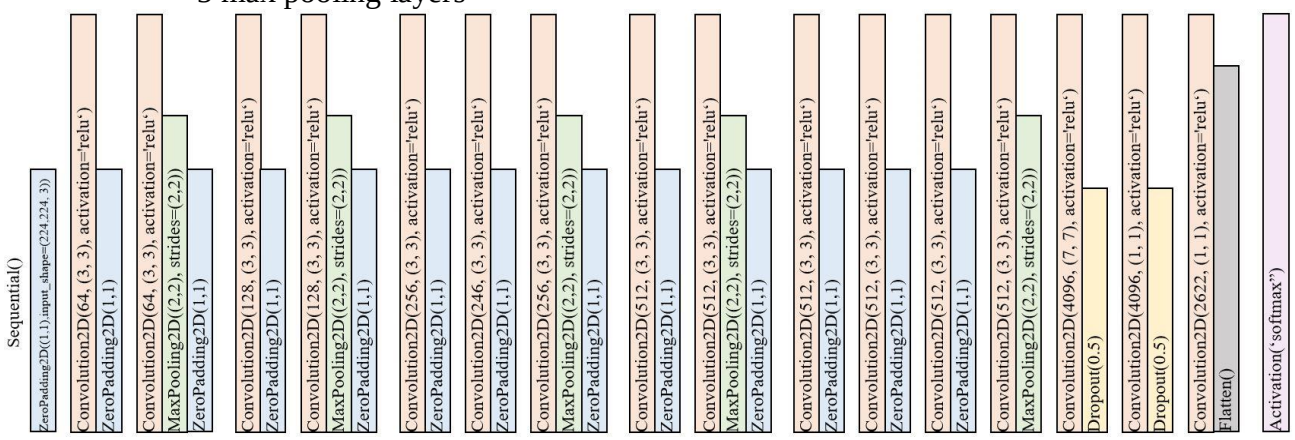


Figure 3.4: *VGG16 model architecture with details*

After defining the model as it is, prepared the training and test data by encoding it using this model, the results are four numpy arrays files and a weight file (train data, train labels, test data and test labels) we than removed the last Softmax layer and added these layers:

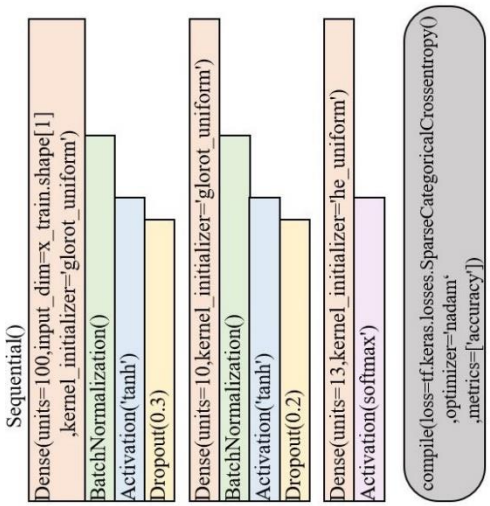


Figure 3.5: *The last layers added to the VGG16 model with their details*

4. Training the model

In short, train the model on the training data and validate it on the test data numpy array files. Once we are satisfied with the model's performance on the validation set, we can use it for making predictions on new unknown data.

In this stage the number of epochs is defined, we tried many choices and settled with 50 epochs which (epochs are the number of how many times the model training is repeated), it is usually done with a shuffle and random choices of the dataset items. After, we saved the model for later use.

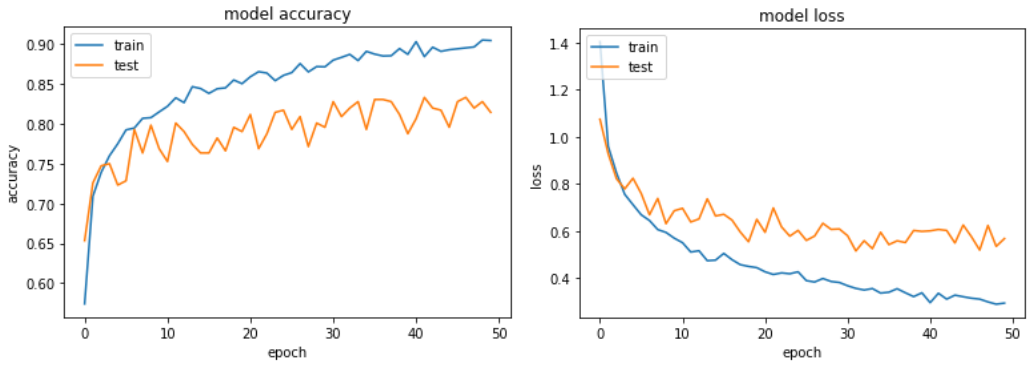


Figure 3.6: *The accuracy and loss values in the fitting task*

5. Estimating the model's performance

Finally, we load the test data (images) and go through the pre-processing step here as well. We then predict the classes for these images using the trained model and estimate how well the model perform by comparing how many correct predictions and mistakes (error) were made.

***Figure 3.7**: Some of the result retrieved on new unknown images.*

The model achieved great results with accuracy over 99% in almost all the new unknown test images which is considered to be expected from a strong model such as VGG16, although it's outdated compared to newer models but still outperforms all of them in the field of Face Recognition.

6. Conclusion

In this Chapter, we have presented the conception with its implementation details, starting from the main conception graph, we divided the work into four stages as mentioned in the introduction, moving on we provided some real face images examples from the dataset that we made with their preprocessing results. Next, we used the model VGG16 for face recognition specifying each of its layer's parameters such as data size and types and also the activation functions used.

Then we added the specification of the layers added after removing the last softmax layer from VGG16, for transfer learning using Batch Normalization, dense and other activation functions. Last, we compiled the new model using Adam optimizer and Keras's Cross entropy for the calculation of the loss function.

We learned transfer learning, which is a very important way of taking advantage of benchmark pre-trained models which are trained on bigger and richer datasets using faster hardware and for longer periods of times that we can't afford to waste again.

General Conclusion

The implementation of a real-time face recognition application is an undisputed requirement today due to security needs in several areas. Given the quantity of potential software (security, social networks, etc.) that can be based on this application, it must meet the requirements of robustness and speed of results. Our project is a temptation to realize such an application.

In this thesis, we described the problem of 2D face recognition by an algorithm based CNN architecture model, in the presence of illumination variation and poses. The main methods of the literature have been studied, whether on face recognition systems or the various works on deep learning. And we particularly focused on choosing the best convolutional neural network and the best method of loss reduction and performance improvement in a complex environment.

The method we used is a very powerful convolutional neural network which is the VGG for face recognition specifying each of its layer's parameters such as data size and types and also the activation functions used, the model obtained was tested on the database that we created and augmented to test, we added the specification of the layers added after removing the last softmax layer from VGG16, for transfer learning using Batch Normalization, dense and other activation functions. Last, we compiled the new model using Adam optimizer and Keras's Cross entropy for the calculation of the loss function. the results obtained allowed us to make a comparison with some existing work, which has given us very good results.

Finally, before moving on to perspectives, this work allowed us to put into practice our knowledge of neural networks and to acquire others and the time spent reading articles served as a good introduction to the research.

Based on the performance results obtained, the following perspectives can be proposed:

- ✓ Try other architectures: ResNet, Inception, Xecption, DensNet ...
- ✓ Make an embedded version on mobile, or Rasberrypi.
- ✓ Train the model on a very large database such as VGG face 2.
- ✓ Do a Detection combined with recognition using a YOLO or SSD model.

References

[1]. Turk, M. and Pentland, A., (1991). Eigenfaces for recognition. Journal of cognitive neuroscience, 3(1), pp.71-86.

[2]. Bouzit Dhikra and Hallaci Samir « Reconnaissance de visage basée sur une approche triangulaire » page 8_26 Mémoire Master informatique,Université 8 mai 1945 Guelma, 2019.

[3]. https://goldenmeancalipers.com/2011/12/phi-and-the-human-face/ last visit 08/09/2020

[4]. Chellappa, R., Wilson, C.L. and Sirohey, S., (1995). Human and machine recognition of faces: A survey. Proceedings of the IEEE, 83(5), pp.705-741.

[5]. R. Brunelli, R. and Poggio, T., (1993). Face recognition: Features versus templates. IEEE transactions on pattern analysis and machine intelligence, 15(10), pp.1042-1052.

[6]. Alpaydin, Ethem (2010). Introduction to Machine Learning. MIT Press. p. 9.

[7]. P. Parveen and B. Thuraisingham, (2006) "Face Recognition Using Multiple Classifiers," 2006 18th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'06), Arlington, VA , pp. 179-186.

[8]. Wang, M. and Deng, W. (2018), Deep face recognition: A survey. arXiv preprint arXiv:1804.06655.

[9]. LeCun, Y., Bengio, Y. and Hinton, G., (2015). Deep learning. nature, 521(7553), pp.436-444.

[10]. Mitchell, T.M., 1997. Machine learning. (1997). Burr Ridge, IL: McGraw Hill, 45(37), pp.870-877.

[11]. https://wordstream-files-prod.s3.amazonaws.com/s3fs-public/machine-learning.png last access 08/05/2020

[12]. Russell, S.J. and Norvig, P., (2010). Artificial Intelligence-A Modern Approach, Third International Edition.

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

[14]. Glorot, X. and Bengio, Y., (2010), March. Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the thirteenth international conference on artificial intelligence and statistics (pp. 249-256).

[15]. Kohavi, R. and Wolpert, D.H., (1996), July. Bias plus variance decomposition for zero-one loss functions. In ICML (Vol. 96, pp. 275-83).

[16]. Zhu, X., Vondrick, C., Fowlkes, C.C. and Ramanan, D., 2016. Do we need more training data?. International Journal of Computer Vision, 119(1), pp.76-92. arXiv:1503.01508

[17]. Prajit Ramachandran, Barret Zoph, Quoc V. Le, Searching for activation functions (2017), arXiv:1710.05941

[18]. Ramachandran, P., Zoph, B. and Le, Q.V., (2017). Searching for activation functions. arXiv preprint arXiv:1710.05941.

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

[20]. https://cs231n.github.io/convolutional-networks/, last access 02/05/2020

[21]. http://www.deeplearningbook.org/contents/rnn.html, last access 04/05/2020

[22]. Hochreiter, S. and Schmidhuber, J., 1997. Long short-term memory. Neural computation, 9(8), pp.1735-1780.

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

[24]. Schmidhuber, J. (2015). "Deep Learning in Neural Networks: An Overview". Neural Networks. 61: 85–117. arXiv:1404.7828. doi:10.1016/j.neunet.2014.09.003. PMID 25462637. S2CID 11715509.

[25]. Schmidhuber, J., 2015. Deep learning in neural networks: An overview. Neural networks, 61, pp.85-117.

[26]. LeCun, Y., Bengio, Y. and Hinton, G., 2015. Deep learning. nature, 521(7553), pp.436-444.

[27]. Masi, I., Wu, Y., Hassner, T. and Natarajan, P., (2018), October. Deep face recognition: A survey. In 2018 31st SIBGRAPI conference on graphics, patterns and images (SIBGRAPI) (pp. 471-478). IEEE.

[28]. Hu, G., Yang, Y., Yi, D., Kittler, J., Christmas, W., Li, S.Z. and Hospedales, T., (2015). When face recognition meets with deep learning: an evaluation of convolutional neural networks for face recognition. In Proceedings of the IEEE international conference on computer vision workshops (pp. 142-150).

[29]. Chaudhuri, A., (2020). Deep Learning Models for Face Recognition: A Comparative Analysis. In Deep Biometrics (pp. 99-140). Springer, Cham.

[30]. Shepley, A.J., (2019). Deep Learning For Face Recognition: A Critical Analysis. arXiv preprint arXiv:1907.12739.

[31]. C. Szegedy, W. Liu, Y. Jia, et al (2014) “Going deeper with convolutions,” arXiv preprint arXiv:1409.4842

[32]. P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. (2014) Overfeat: Integrated recognition, localization and detection using convolutional networks. In ICLR

[33]. Goodfellow, I., (2014). Pouget‐Abadie J, Mirza M, et al. Generative Adversarial Nets, 2672, p.2680.

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

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

[36]. Girshick, R., Donahue, J., Darrell, T. and Malik, J., (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 580-587).

[37]. Ioffe, S. and Szegedy, C., (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167.

[38]. He, K., Zhang, X., Ren, S. and Sun, J., (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778). arXiv preprint arXiv:1512.03385.

[39]. Badrinarayanan, V., Handa, A. and Cipolla, R., (2015). Segnet: A deep convolutional encoder-decoder architecture for robust semantic pixel-wise labelling. arXiv preprint arXiv:1505.07293.

[40]. Badrinarayanan, V., Kendall, A. and Cipolla, R., (2017). Segnet: A deep convolutional encoder-decoder architecture for image segmentation. IEEE transactions on pattern analysis and machine intelligence, 39(12), pp.2481-2495. arXiv preprint arXiv:1511.00561

[41]. Girshick, R., (2015). Fast r-cnn. In Proceedings of the IEEE international conference on computer vision (pp. 1440-1448). arXiv:1504.08083

[42]. Ren, S., He, K., Girshick, R. and Sun, J., (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In Advances in neural information processing systems (pp. 91-99).

[43]. Srivastava, R.K., Greff, K. and Schmidhuber, J., (2015). Training very deep networks. In Advances in neural information processing systems (pp. 2377-2385).arXiv preprint arXiv:1507.06228v2

[44]. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. and Wojna, Z., (2016). Rethinking the inception architecture for computer vision. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2818-2826).arXiv:1512.00567

[45]. Szegedy, C., Ioffe, S., Vanhoucke, V. and Alemi, A., (2016). Inception-v4, inception-resnet and the impact of residual connections on learning. arXiv preprint arXiv:1602.07261.

[46]. He, K., Zhang, X., Ren, S. and Sun, J., (2016), October. Identity mappings in deep residual networks. In European conference on computer vision (pp. 630-645). Springer, Cham. arXiv preprint arXiv:1603.05027v3

[47]. Xie, S., Girshick, R., Dollár, P., Tu, Z. and He, K., (2017). Aggregated residual transformations for deep neural networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1492-1500). arXiv preprint arXiv:1611.05431v1

[48]. Paszke, A., Chaurasia, A., Kim, S. and Culurciello, E., (2016). Enet: A deep neural network architecture for real-time semantic segmentation. arXiv preprint arXiv:1606.02147.

[49]. Iandola, F.N., Han, S., Moskewicz, M.W., Ashraf, K., Dally, W.J. and Keutzer, K., (2016). SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size. arXiv preprint arXiv:1602.07360.

[50]. Redmon, J., Divvala, S., Girshick, R. and Farhadi, A., (2016). You only look once: Unified, real-time object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 779-788).

[51]. Chollet, F., (2017). Xception: Deep learning with depthwise separable convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1251-1258). arXiv preprint arXiv:1610.02357

[52]. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y. and Berg, A.C., (2016), October. Ssd: Single shot multibox detector. In European conference on computer vision (pp. 21-37). Springer, Cham.

[53]. Huang, G., Liu, Z., Van Der Maaten, L. and Weinberger, K.Q., (2017). Densely connected convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4700-4708).

[54]. Hasanpour, S.H., Rouhani, M., Fayyaz, M. and Sabokrou, M., (2016). Lets keep it simple, using simple architectures to outperform deeper and more complex architectures. arXiv preprint arXiv:1608.06037.

[55]. Sabour, S., Frosst, N. and Hinton, G.E., 2017. Dynamic routing between capsules. In Advances in neural information processing systems (pp. 3856-3866).

[56]. Redmon, J. and Farhadi, A., 2017. YOLO9000: better, faster, stronger. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 7263-7271).

[57]. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M. and Adam, H., 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861.

[58]. Zhang, S., Wen, L., Bian, X., Lei, Z. and Li, S.Z., 2018. Single-shot refinement neural network for object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4203-4212).arXiv preprint arXiv:1711.06897

[59]. Lin, T.Y., Goyal, P., Girshick, R., He, K. and Dollár, P., 2017. Focal loss for dense object detection. In Proceedings of the IEEE international conference on computer vision (pp. 2980-2988). arXiv preprint arXiv:1708.02002.

[60]. Dai, J., Qi, H., Xiong, Y., Li, Y., Zhang, G., Hu, H. and Wei, Y., 2017. Deformable convolutional networks. In Proceedings of the IEEE international conference on computer vision (pp. 764-773).

[61]. Redmon, J. and Farhadi, A., 2018. Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767.

[62]. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A. and Chen, L.C., 2018. Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4510-4520).

[63]. Wong, A., Famuori, M., Shafiee, M.J., Li, F., Chwyl, B. et al 2019. YOLO nano: A highly compact you only look once convolutional neural network for object detection. arXiv preprint arXiv:1910.01271.

[64]. Kaiming He et Al (2014), Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition, arXiv:1406.4729

[65]. Kaiming He et Al (2015), Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification, arXiv:1502.01852

[66]. Redmon, J et Al (2017), YOLO9000: Better, Faster, Stronger, arXiv:1612.08242

[67]. Sergey Z et Al (2016), Wide Residual Networks, arXiv:1605.07146

[68]. Dongyoon H et Al (2017), Deep Pyramidal Residual Networks, arXiv:1610.02915

[69]. Gustav L et Al (2016), FractalNet: Ultra-Deep Neural Networks without Residuals, arXiv:1605.07648

[70]. Xiangyu Z et Al (2017), ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices, arXiv:1707.01083

[71]. Jie Hu, Li Shen et Al (2017), Squeeze-and-Excitation Networks, arXiv:1709.01507

[72]. Rranav Rajpurkar, Jeremy Irvin et Al (2017), CheXNet: Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning, arXiv:1711.05225

[73]. Yunpeng C et Al (2017), Dual Path Networks, arXiv:1707.01629

[74]. Aidan N et Al (2017), The Reversible Residual Network: Backpropagation Without Storing Activations, arXiv:1707.04585

[75]. Joachim D et Al (2017), McKernel: A Library for Approximate Kernel Expansions in Log-linear Time, arXiv:1702.08159

[76]. Mingxing T et Al (2018), MnasNet: Platform-Aware Neural Architecture Search for Mobile, arXiv:1807.11626

[77]. Ninging M et Al (2018), ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design, arXiv:1807.11164

[78]. Esteban R et Al (2018), Regularized Evolution for Image Classifier Architecture Search, arXiv:1802.01548

[79]. Zeming L, Chao P et Al (2018), DetNet: A Backbone network for Object Detection, arXiv:1804.06215

[80]. Robert J. Wang et Al (2018), Pelee: A Real-Time Object Detection System on Mobile Devices,

[81]. Bichen W et Al (2018), FBNet: Hardware-Aware Efficient ConvNet Design via Differentiable Neural Architecture Search, arXiv:1812.03443

[82]. Amir Gholami, Kiseok K et Al (2018), SqueezeNext: Hardware-Aware Neural Network Design, arXiv:1803.10615

[83]. Han C. et Al (2018), ProxylessNAS: Direct Neural Architecture Search on Target Task and Hardware, arXiv:1812.00332

[84]. Huiyu W et Al (2018), ELASTIC: Improving CNNs with Dynamic Scaling Policies, arXiv:1812.05262

[85]. Tong He, Zhi Z et Al (2018), Bag of Tricks for Image Classification with Convolutional Neural Networks, arXiv:1812.01187

[86]. Sachin M et Al (2018), ESPNetv2: A Light-weight, Power Efficient, and General Purpose Convolutional Neural Network, arXiv:1811.11431

[87]. Chun-Fu C, Quanfu F et Al (2018), Big-Little Net: An Efficient Multi-Scale Feature Representation for Visual and Speech Recognition, arXiv:1807.03848

[88]. Mingxing Tan, Quoc V. Le (2019), EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks, arXiv:1905.11946

[89]. Jingdong W et Al (2019), Deep High-Resolution Representation Learning for Visual Recognition, arXiv:1908.07919

[90]. Andrew W, Mark S et Al (2019), Searching for MobileNetV3, arXiv:1905.02244

[91]. Dimitrios S, Ruizhou D et Al (2019), Single-Path NAS: Designing Hardware-Efficient ConvNets in less than 4 Hours, arXiv:1904.02877

[92]. Zheng Q, Zeming L et Al (2019), ThunderNet: Towards Real-time Generic Object Detection, arXiv: 1903.11752

[93]. Mingxing Tan, Quoc V. Le (2019), MixConv: Mixed Depthwise Convolutional Kernels, arXiv: 1907.09595

[94]. Xianzhi D et al (2019), SpineNet: Learning Scale-Permuted Backbone for Recognition and Localization, arXiv:1912.05027

[95]. Youngwan L et al (2019), An Energy and GPU-Computation Efficient Backbone Network for Real-Time Object Detection, arXiv:1904.09730

[96]. Chien-Yao W et al (2019), CSPNet: A New Backbone that can Enhance Learning Capability of CNN, arXiv:1911.11929

[97]. Youngwan L, Jongyoul P (2019), CenterMask : Real-Time Anchor-Free Instance Segmentation, arXiv:1911.06667

[98]. Jiemin F et al (2019), Densely Connected Search Space for More Flexible Neural Architecture Search, arXiv:1906.09607

[99]. Yi L et al (2019), Data-Driven Neuron Allocation for Scale Aggregation Networks, arXiv:1904.09460

[100]. Sachin M et al (2019), DiCENet: Dimension-wise Convolutions for Efficient Networks, arXiv:1906.03516

[101]. Xianxiang C et al (2019), MoGA: Searching Beyond MobileNetV3, arXiv:1908.01314

[102]. Qilong W et al (2019), ECA-Net: Efficient Channel Attention for Deep Convolutional Neural Networks, arXiv:1910.03151

[103]. Kai H et al (2019), GhostNet: More Features from Cheap Operations, arXiv:1911.11907

[104]. Jiemin F et al (2019), Densely Connected Search Space for More Flexible Neural Architecture Search, arXiv:1906.09607

[105]. Maxim B, Hervé J et al (2019), MultiGrain: a unified image embedding for classes and instances, arXiv:1902.05509

[106]. Saining X et al (2019), Exploring Randomly Wired Neural Networks for Image Recognition, arXiv:1904.01569

[107]. Xiang L et al (2019), Selective Kernel Networks, Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 510-519

[108]. Xiangxiang C et al (2019), SCARLET-NAS: Bridging the gap between Stability and Scalability in Weight-sharing Neural Architecture Search, arXiv:1908.06022

[109]. Yukang C et al (2019), DetNAS: Backbone Search for Object Detection, arXiv:1903.10979

[110]. Alexey B et al (2020), YOLOv4: Optimal Speed and Accuracy of Object Detection, arXiv:2004.1093

[111]. Zahra Noury, Mahdi Rezaei (2020), Deep-CAPTCHA: a deep learning based CAPTCHA solver for vulnerability assessment, arXiv:2006.08296

[112]. Ilija R et al (2020), Designing Network Design Spaces, arXiv:2003.13678

[113]. Jungkyu L et al (2020), Compounding the Performance Improvements of Assembled Techniques in a Convolutional Neural Network, arXiv:2001.06268

[114]. Tal R et al (2020), TResNet: High Performance GPU-Dedicated Architecture, arXiv:2003.13630

[115]. Hang Z, Chongruo W et al (2020), ResNeSt : Split-Attention Networks, arXiv:2004.08955

[116]. Shan Y et al (2020), GreedyNAS: Towards Fast One-Shot NAS with Greedy Supernet, arXiv:2003.11236

[117]. Matej U et al (2020), Harmonic Convolutional Networks based on Discrete Cosine Transform, arXiv:2001.06570

[118]. François-Lavet et al (2018). "An Introduction to Deep Reinforcement Learning". Foundations and Trends in Machine Learning. 11 (3–4): 219–354. arXiv:1811.12560.

[119]. Mengja Y et al (2019), VarGFaceNet: An Efficient Variable Group Convolutional Neural Network for Lightweight Face Recognition, arXiv:1910.04985

[120]. Jiankang D et al (2019), ArcFace: Additive Angular Margin Loss for Deep Face Recognition, arXiv:1801.07698

[121]. Yichun S et al (2019), Probabilistic Face Embeddings, arXiv:1904.09658

[122]. Qian Z et al (2020),VarGNet: Variable Group Convolutional Neural Network for Efficient Embedded Computing,arXiv:1907.05653

[123]. Hao W et al (2018), CosFace: Large Margin Cosine Loss for Deep Face Recognition,arXiv:1801.09414

[124]. Xiao Z et al (2019), AdaCos: Adaptively Scaling Cosine Logits for Effectively Learning Deep Face Representations, arXiv:1905.00292

[125]. Hung, S.C., Lee, J.H., Wan, T.S., Chen, C.H., Chan, Y.M. and Chen, C.S., (2019), June. Increasingly packing multiple facial-informatics modules in a unified deep-learning model via lifelong learning. In Proceedings of the 2019 on International Conference on Multimedia Retrieval (pp. 339-343).

[126]. Jintao Z, DiDiChuxing (2019), SeesawFaceNets: sparse and robust face verification model for mobile platform, arXiv:1908.09124

[127]. Florian S et al (2015), FaceNet: A Unified Embedding for Face Recognition and Clustering, arXiv:1503.03832

[128]. Yi Sun et al (2015), DeepID3: Face Recognition with Very Deep Neural Networks, arXiv:1502.00873

[129]. Yutong Z et al (2018), Ring loss: Convex Feature Normalization for Face Recognition,arXiv:1803.00130

[130]. Yi S, Xiaogang W et al (2014), Deeply learned face representations are sparse, selective, and robust ,arXiv:1412.1265

[131]. Weiyang L et al (2018), SphereFace: Deep Hypersphere Embedding for Face Recognition ,arXiv:1704.08063

[132]. Xiang W et al (2018), A Light CNN for Deep Face Representation with Noisy Labels,arXiv:1511.02683

[133]. Alessandro C et al (2018), Git Loss for Deep Face Recognition,arXiv:1807.08512

[134]. Steven C et al (2019), Compacting, Picking and Growing for Unforgetting Continual Learning ,arXiv:1910.06562

[135]. Zuheng M et al (2019), Dynamic Multi-Task Learning for Face Recognition with Facial Expression ,arXiv:1911.03281

[136]. Yi S et al (2014), Deep Learning Face Representation by Joint Identification-Verification,arXiv:1406.4773

[137]. Wei H et al (2018), SeqFace: Make full use of sequence information for face recognition,arXiv:1803.06524

[138]. Xin L et al (2016), VIPLFaceNet: An Open Source Deep Face Recognition SDK,arXiv:1609.03892

[139]. Chaochao L et al (2014), Surpassing Human-Level Face Verification Performance on LFW with GaussianFace,arXiv:1404.3840

[140]. Baris G et al (2018), Semi-supervised Adversarial Learning to Generate Photorealistic Face Images of New Identities from 3D Morphable Model,arXiv:1804.03675

[141]. Anh T et al (2016), Regressing Robust and Discriminative 3D Morphable Models with a very Deep Neural Network,arXiv:1612.04904

[142]. Mather, G., (2012). The visual cortex. School of Life Sciences: University of Sussex. University of Sussex.

[143]. Zell, Andreas (2003). "chapter 5.2". Simulation neuronaler Netze [Simulation of Neural Networks] (in German) (1st ed.). Addison-Wesley. ISBN 978-3-89319-554-1. OCLC 249017987.

[144]. The Neural Network Zoo | Stefan Leijnen and Fjodor van Veen | Research Gate | https://www.researchgate.net/publication/341373030_The_Neural_Network_Zoo last access 20/08/2020

[145]. LeCun, Y et al (1990). Handwritten digit recognition with a back-propagation network.Advances in Neural Information Processing Systems, 2, 396–404, Morgan Kaufman.

[146]. Belhumeur, P.N., Hespanha, J.P. and Kriegman, D.J., (1997). Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. IEEE Transactions on pattern analysis and machine intelligence, 19(7), pp.711-720.

[147]. Phillips, P.J., Wechsler, H., Huang, J. and Rauss, P.J., (1998). The FERET database and evaluation procedure for face-recognition algorithms. Image and vision computing, 16(5), pp.295-306.

[148]. Sim, T., (2003). S. Baker, and M. Bsat. The cmu pose, illumination, and expression (pie) database. IEEE Transactions on Pattern Analysis and Machine Intelligence, 25(12), pp.1615-1618.

[149]. Gross, R., Matthews, I., Cohn, J., Kanade, T. and Baker, S., (2010). Multi-pie. Image and Vision Computing, 28(5), pp.807-813.

[150]. http://vision.ucsd.edu/~iskwak/ExtYaleDatabase/ExtYaleB.html 02/04/2020

[151]. Gao, W., Cao, B., Shan, S., Chen, X., Zhou, D., Zhang, X. and Zhao, D., (2007). The CAS-PEAL large-scale Chinese face database and baseline evaluations. IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, 38(1), pp.149-161.

[152]. Goh, R., Liu, L., Liu, X. and Chen, T., 2005, October. The CMU face in action (FIA) database. In International Workshop on Analysis and Modeling of Faces and Gestures (pp. 255-263). Springer, Berlin, Heidelberg.

[153]. Liu, Z., Luo, P., Wang, X. and Tang, X., 2018. Large-scale celebfaces attributes (celeba) dataset. Retrieved August, 15, p.2018.

[154]. Huang, G.B., Mattar, M., Berg, T. and Learned-Miller, E., 2008, October. Labeled faces in the wild: A database forstudying face recognition in unconstrained environments.

[155]. Grgic, M., Delac, K. and Grgic, S.,(2011). SCface–surveillance cameras face database. Multimedia tools and applications, 51(3), pp.863-879.

[156]. Lei L et al (2019), 3D Face Mask Presentation Attack Detection Based on Intrinsic Image Analysis, arXiv:1903.11303

[157]. Karras, T., Laine, S. and Aila, T., 2019. A style-based generator architecture for generative adversarial networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4401-4410).

[158]. A. M. Martinez and R. Benavente.(1998) The AR face database. Technical Report 24, Computer Vision Center, University of Barcelona,

[159]. Kleber, F., Fiel, S., Diem, M. and Sablatnig, R., 2013, August. Cvl-database: An off-line database for writer retrieval, writer identification and word spotting. In 2013 12th international conference on document analysis and recognition (pp. 560-564). IEEE.

[160]. Milborrow, S., Morkel, J. and Nicolls, F., 2010. The MUCT landmarked face database. Pattern Recognition Association of South Africa, 201(0).

[161]. Geralds, J (2017), UTKFace Large Scale Face Dataset. github. com.

[162]. Florian, R., Ittycheriah, A., Jing, H. and Zhang, T., 2003. Named entity recognition through classifier combination. In Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003 (pp. 168-171).

[163]. Salimans, T. and Kingma, D.P., 2016. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In Advances in neural information processing systems (pp. 901-909).

[164]. Ba, J.L., Kiros, J.R. and Hinton, G.E., 2016. Layer normalization. arXiv preprint arXiv:1607.06450.

[165]. Ioffe, S. and Szegedy, C., 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167.

[166]. Krizhevsky, A., Sutskever, I. and Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp. 1097-1105).

[167]. Ulyanov, D., Vedaldi, A. and Lempitsky, V., 2016. Instance normalization: The missing ingredient for fast stylization. arXiv preprint arXiv:1607.08022.

[168]. Miyato, T., Kataoka, T., Koyama, M. and Yoshida, Y., 2018. Spectral normalization for generative adversarial networks. arXiv preprint arXiv:1802.05957.

[169]. Ogasawara, E., Martinez, L.C., De Oliveira, D., Zimbrão, G., Pappa, G.L. and Mattoso, M., 2010, July. Adaptive normalization: a novel data normalization approach for non-stationary time series. In The 2010 International Joint Conference on Neural Networks (IJCNN) (pp. 1-8). IEEE.

[170]. Salimans, T. and Kingma, D.P., 2016. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In Advances in neural information processing systems (pp. 901-909).

[171]. De Vries, H., Strub, F., Mary, J., Larochelle, H., Pietquin, O. and Courville, A.C., 2017. Modulating early visual processing by language. In Advances in Neural Information Processing Systems (pp. 6594-6604).

[172]. Wu, Y. and He, K., 2018. Group normalization. In Proceedings of the European conference on computer vision (ECCV) (pp. 3-19).

[173]. Kingma, D.P. and Dhariwal, P., 2018. Glow: Generative flow with invertible 1x1 convolutions. In Advances in neural information processing systems (pp. 10215-10224).

[174]. Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J. and Aila, T., 2020. Analyzing and improving the image quality of stylegan. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 8110-8119).

[175]. Luo, P., Ren, J., Peng, Z., Zhang, R. and Li, J., 2018. Differentiable learning-to-normalize via switchable normalization. arXiv preprint arXiv:1806.10779.

[176]. Foracchia, M., Grisan, E. and Ruggeri, A., 2005. Luminosity and contrast normalization in retinal images. Medical Image Analysis, 9(3), pp.179-190.

[177]. Qiao, S., Wang, H., Liu, C., Shen, W. and Yuille, A., 2019. Weight standardization. arXiv preprint arXiv:1903.10520.

[178]. Huang, X. and Belongie, S., 2017. Arbitrary style transfer in real-time with adaptive instance normalization. In Proceedings of the IEEE International Conference on Computer Vision (pp. 1501-1510).

[179]. He, K., Girshick, R. and Dollár, P., 2019. Rethinking imagenet pre-training. In Proceedings of the IEEE international conference on computer vision (pp. 4918-4927).

[180]. Li, X., Sun, W. and Wu, T., 2019. Attentive normalization. arXiv preprint arXiv:1908.01259.

[181]. Huang, L., Yang, D., Lang, B. and Deng, J., 2018. Decorrelated batch normalization. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 791-800).

[182]. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R., 2014. Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research, 15(1), pp.1929-1958.

[183]. Loshchilov, I. and Hutter, F., 2017. Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101.

[184]. Choe, J. and Shim, H., 2019. Attention-based dropout layer for weakly supervised object localization. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 2219-2228).

[185]. Pereyra, G., Tucker, G., Chorowski, J., Kaiser, Ł. and Hinton, G., 2017. Regularizing neural networks by penalizing confident output distributions. arXiv preprint arXiv:1701.06548.

[186]. Grandvalet, Y. and Bengio, Y., 2006. Entropy Regularization.

[187]. Yao, Y., Rosasco, L. and Caponnetto, A., 2007. On early stopping in gradient descent learning. Constructive Approximation, 26(2), pp.289-315.

[188]. Kingma, D.P., Salimans, T. and Welling, M., 2015. Variational dropout and the local reparameterization trick. In Advances in neural information processing systems (pp. 2575-2583).

[189]. Wan, L., Zeiler, M., Zhang, S., Le Cun, Y. and Fergus, R., 2013, February. Regularization of neural networks using dropconnect. In International conference on machine learning (pp. 1058-1066).

[190]. Elden, L., 1977. Algorithms for the regularization of ill-conditioned least squares problems. BIT Numerical Mathematics, 17(2), pp.134-145.

[191]. Park, M.Y. and Hastie, T., 2007. L1-regularization path algorithm for generalized linear models. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 69(4), pp.659-677.

[192]. Gal, Y. and Ghahramani, Z., 2016. A theoretically grounded application of dropout in recurrent neural networks. In Advances in neural information processing systems (pp. 1019-1027).

[193]. Kim, S.J., Koh, K., Lustig, M., Boyd, S. and Gorinevsky, D., 2007. An interior-point method for large-scale $\ell_1 $-regularized least squares. IEEE journal of selected topics in signal processing, 1(4), pp.606-617.

[194]. Oster, H.S. and Rudy, Y., 1992. The use of temporal information in the regularization of the inverse problem of electrocardiography. IEEE transactions on biomedical engineering, 39(1), pp.65-75.

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

[196]. Robbins, H. and Monro, S., 1951. A stochastic approximation method. The annals of mathematical statistics, pp.400-407..

[197]. Boyd, S., Parikh, N. and Chu, E., 2011. Distributed optimization and statistical learning via the alternating direction method of multipliers. Now Publishers Inc.

[198]. Geoffrey Hinton Neural Networks for machine learning online course, RMSprop-Optimization algorithms, https://www.cs.toronto.edu/~hinton/nntut.html last access 06/08/2020

[199]. Duchi, J., Hazan, E. and Singer, Y., 2011. Adaptive subgradient methods for online learning and stochastic optimization. Journal of machine learning research, 12(7).

[200]. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B. and Hochreiter, S., 2017. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In Advances in neural information processing systems (pp. 6626-6637).

[201]. Pascanu, R., Mikolov, T. and Bengio, Y., 2013, February. On the difficulty of training recurrent neural networks. In International conference on machine learning (pp. 1310-1318).

[202]. You, Y., et al 2019. Large batch optimization for deep learning: Training bert in 76 minutes. arXiv preprint arXiv:1904.00962.

[203]. Reddi, S.J., Kale, S. and Kumar, S., 2019. On the convergence of adam and beyond. arXiv preprint arXiv:1904.09237.

[204]. Nesterov, Y. (1983). A method for solving a convex programming problem with convergence rate O(1/k 2). Soviet Math. Dokl., 27, 372--376.

[205]. Shazeer, N. and Stern, M., 2018. Adafactor: Adaptive learning rates with sublinear memory cost. arXiv preprint arXiv:1804.04235.

[206]. Rattray, M., Saad, D. and Amari, S.I., 1998. Natural gradient descent for on-line learning. Physical review letters, 81(24), p.5461.

[207]. You, Y., Gitman, I. and Ginsburg, B., 2017. Large batch training of convolutional networks. arXiv preprint arXiv:1708.03888.

[208]. Jaderberg, M., et al 2017. Population based training of neural networks. arXiv preprint arXiv:1711.09846.

[209]. Abadi, M., et al 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. arXiv preprint arXiv:1603.04467.

[210]. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L. and Lerer, A., 2017. Automatic differentiation in pytorch.

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

[212]. Seide, F. and Agarwal, A., 2016, August. CNTK: Microsoft's open-source deep-learning toolkit. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 2135-2135).

[213]. Lin, W.F., Tsai, D.Y., Tang, L., Hsieh, C.T., Chou, C.Y., Chang, P.H. and Hsu, L., 2019, March. ONNC: A compilation framework connecting ONNX to proprietary deep learning accelerators. In 2019 IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS) (pp. 214-218). IEEE.

[214]. Chen, T., Li, M., Li, Y., Lin, M., Wang, N., Wang, M., Xiao, T., Xu, B., Zhang, C. and Zhang, Z., 2015. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. arXiv preprint arXiv:1512.01274.

[215]. Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S. and Darrell, T., 2014, November. Caffe: Convolutional architecture for fast feature embedding. In Proceedings of the 22nd ACM international conference on Multimedia (pp. 675-678).

[216]. Q. Cao, L. Shen, W. Xie, O. M. Parkhi, A. Zisserman, VGGFace2: A dataset for recognising faces across pose and age. Arxiv: https://arxiv.org/abs/1710.08092.

[217]. Dong Y et al, “Learning Face Representation from Scratch”. arXiv preprint arXiv:1411.7923. 2014.

[218]. Yi, D., Lei, Z., Liao, S. and Li, S.Z., 2014. Learning face representation from scratch. arXiv preprint arXiv:1411.7923.

[219]. Lee, C.H., Liu, Z., Wu, L. and Luo, P., 2020. Maskgan: Towards diverse and interactive facial image manipulation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 5549-5558).

[220]. Jain, V. and Learned-Miller, E., 2010. Fddb: A benchmark for face detection in unconstrained settings (Vol. 2, No. 4, p. 5). UMass Amherst technical report.

[221]. Yang, Shuo and Luo, Ping and Loy, Chen Change and Tang, Xiaoou.(2016) WIDER FACE: A Face Detection Benchmark,(CVPR), arXiv:1511.06523

[222]. Rothe, R., Timofte, R. and Van Gool, L., 2015. Dex: Deep expectation of apparent age from a single image. In Proceedings of the IEEE international conference on computer vision workshops (pp. 10-15).