DEEP LEARNING BASED FACE ATTRIBUTES RECOGNITION

MOHAMAD HAZIM BIN SAIDI

UNIVERSITI SAINS MALAYSIA

2018

ACKNOWLEDGEMENT

First of all, I would like to thank Prof. Madya Dr. Shahrel Azmin Bin Suandi for his inspiration and guidance for my final year project (FYP). The decision to take up Deep Learning based project is quite tough, but with his encouragement, this project is successfully completed. Discovery and exploration into the realm of Deep Learning is the greatest experience from this project, shall never be forgotten.

A big commendation for Face Recognition Homepage for providing free facial database. A special thanks to the creator and developer of SCFace, SoF, IMFDB, LFW, Fei and GT database for providing me a variety of facial images.

Special appreciation to administration officers of School of Electrical and Electronic Engineering (PPKEE) for arranging numerous talks and workshop in order to prepare and guide me and other students throughout this year. Not to forget about the academic staffs and lab assistant that provide generous advices and technical support for me to complete the project. Special thanks to Encik Abdul Latif for providing easy accessibility to computer in Communication Lab. The hardware is a crucial component in my project and hence her help is much appreciated.

Besides, I am extremely thankful to the develop team behind Tensorflow whom open-source this tool for public and allows anyone to be part of Deep Learning research community. It provides sufficient tutorial for me to finish up my project with Fast CNN architecture. Without them, the project would be much harder to proceed.

Last but not least, I would like to extend my gratitude to my family and friends whom supported me during FYP period and had given useful advice and positive encouragement when I was down.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AdaGrad	Adaptive Subgradient Descent
Adam	Adaptive Moment Estimate
AI	Artificial Intelligence
ANN	Artificial Neural Network
API	Application Program Interface
CNN	Convolution Neural Network
CPU	Central Processing Unit
CV	Computer Vision
DL	Deep Learning
DNN	Deep Neural Network
DRL	Deep Reinforcement Learning
GPU	Graphic Processing Unit
GT	Georgia Tech
IDE	Interactive Development Environment
IMFDB	Indian Movie Face Database
LFW	Labelled Face in the Wild
ML	Machine Learning
MD	Malaysia Database
Nadam	Nesterov-accelerated Adaptive Moment Estimation
NLP	Natural Language Processing
RCNN	Recurrent Convolution Neural Network
ReLu	Rectified Linear Unit
ResNet	Residential Network
RL	Representation Learning

RMSProp	Root Mean Square Propagation
SCFace	Surveillance Camera Face
SoF	Specs on Faces
VGG	Visual Geometry Group

NOTATION

a	A scalar (integer or real)
A	A matrix
A	A tensor
I_n	Identity matrix with *n* rows and *n* columns
$e^{(i)}$	Standard basis vector [0, . . . , 0, 1, 0, . . . , 0] with a 1 at position *i*
$\{0,1,\dots,n\}$	The set of all integers between 0 and *n*
$[a,b]$	The real interval including *a* and *b*
$(a,b]$	The real interval excluding *a* but including *b*
$A_{i,j}$	Element i, j of matrix *A*
$A \odot B$	Element-wise (Hadamard) product of *A* and *B*
$\frac{dy}{dx}$	Derivative of *y* with respect to *x*
$\frac{\partial y}{\partial x}$	Partial derivative of *y* with respect to *x*
$\nabla_x y$	Gradient of *y* with respect to *x*
$\nabla_{\mathrm{X}} y$	Matrix derivatives of *y* with respect to *X*
$\nabla_{\times} y$	Tensor containing derivatives of *y* with respect to X
$\frac{\partial f}{\partial x}$	Jacobian matrix $J \in \mathbb{R}^{m\times n}$ of $f: \mathbb{R}^n \rightarrow \mathbb{R}^m$
$\nabla_x^2 f(x)\ or\ H(f)(x)$	The Hessian matrix of *f* at input point *x*
$a \sim \mathrm{P}$	Random variable a has distribution *P*
$\mathbb{E}_{\mathrm{X}\sim\mathrm{P}}[f(x)]\ or\ \mathbb{E}f(x)$	Expectation of *f*(*x*) with respect to *P*(x)
$f: \mathbb{A} \rightarrow \mathbb{B}$	The function f with domain $\mathbb{A}$ and range $\mathbb{B}$
$f(x;\theta)$	Composition of the functions *f* and *g*
$\lVert x \rVert$	L^2 norm of *x*

p_{data}	The data generating distribution
$\hat{p}_{data}$	The empirical distribution defined by the training set
$\mathbb{X}$	A set of training examples
$x^{(i)}$	The *i*-th example (input) from a dataset
$y^{(i)}$	The target associated with *x*(*i*) for supervised learning

PENGENALAN CIRI-CIRI WAJAH BERASASKAN PEMBELAJARAN MENDALAM

ABSTRAK

Pengenalan Wajah merupakan teknologi yang sedang membangun dengan banyak aplikasi di dalam kehidupan sebenar. Matlamat Projek Tahun Akhir ini adalah untuk mewujudkan Pengenalan Ciri-ciri Wajah yang lengkap untuk keselamatan atau kemudahan. Aplikasi pengenalan wajah secara automatik dapat membantu ahli forensik untuk meninjau sesuatu kawasan dengan adanya Pembelajaran Mesin (ML). Namun in bukan seuatu yang mudah kerana imej yang yang ditangkap adalah sangat berbeza dari segi posisi dan halangan terhadap faktor sukarela dan tidak sukarela. Dengan pengenalan Pembelajaran Mendalam (DL), konsep Rantaian Konvolusi Neural (CNN) yanag dicadangkan pada suatu ketika dahulu akhirnya boleh direalisasikan. Dalam projek ini, Fast CNN model digunakan sebagai teras bagi pengenalan ciri-ciri wajah untuk memabantu pengguna mengenalinya. Pengguna boleh mengenali ciri-ciri wajah termasuk jantina, berkaca mata dan muka berbulu. Pengenalan ciri-ciri wajah juga boleh berfungsi dengan baik dalam mengenali wajah yang dihadapkan dengan sempurna dan tidak sempurna. Optimasi dilakukan secara berperingkat dengan menjalankan eksperimen terhadap parameter untuk latihan rangkain supaya nilai optimum boleh dicapai. Dengen menggunakan model ini, prestasi terbaik boleh dihasilkan dalam mengenali ciri-ciri wajah. Penggabungan algorithm terhadap pengoptimasi memainkankan peranan penting dalm mengoptimasi pembelajaran algorithm. Penambahan lapisan konvolusi juga penting dalam mengekstrak ciri-ciri yang berkaitan dengan imej muka.

DEEP LEARNING BASED FACE ATTRIBUTES RECOGNITION

ABSTRACT

Face Recognition is a recently developing technology with numerous real life applications. The goal of this Final Year Project is to create a complete Face Attributes Recognition for security or facility. The automated face identification application is helpful in assisting forensic to survey an area with the implementation of Machine Learning (ML). It was once a difficult challenge due to uncertainties in the captured such as high variation of pose and obstruction corresponding to voluntary and involuntary factors. With the introduction of Deep Learning (DL), the concept of Convolutional Neural Network (CNN) that was once an idea can be realized. In this project, Fast CNN architecture is used as the core engine to power the face attributes recognition that aims to help users to identify its. The users can identify the face attributes including gender, glasses and facial hair. The face attributes recognition also can be performed well in identifying properly and improperly frontalized faces. Optimization is performed by experimenting in stages with several training parameters to obtain the best value for this unique purpose. Using this architecture, the best performance of training algorithm can be produced in order to recognize face attributes. Combined-algorithm based optimizers plays an important role in optimizing the training algorithm. The addition of convolutional layer is also essential in order to extract related facial features of facial images.

CHAPTER 1

INTRODUCTION

1.1 Background

Face Attributes Recognition is one of the fields from computer vision (CV) that has attracted many interests for a long period. The practical applications for it are numerous including biometrical security to automatically detect facial images (Findling and Mayrhofer, 2012) and fast face detection in violent video scenes (Machaca Arceda et al., 2016). Due to these capabilities, a lot of companies and research centres have been working on it. Usually, the central processing unit (CPU) of a machine was used to train neural networks (Goodfellow et al., 2016h). Nowadays, this method is normally considered as inadequate. Currently, the graphic processing unit (GPU) computing or the CPU of several machines networked together are used instead. Earlier, before these expensive setups are introduced, researchers put too much effort to show that CPU could not cope with the high computational job required by neural networks. It is outside the scope to explain on how to apply efficient numerical CPU code, but it can be highlighted that prudent implementation for particular CPU families can produce great improvements.

The field of soft biometrics was significantly targeted to augment the recognition process by fusion of metrics that were essential to discriminate population rather than individuals (Nixon et al., 2015). Afterward, this was refined to deploy measures that could be applied in discriminating individuals especially using descriptions that can be perceived using human vision and in surveillance imagery. A deep branch of this recent field concerns approaches to estimate soft biometrics using conventional biometrics methods or just from images alone. These three strands merge to form soft biometrics. A functional feature-based approach useful for real forensic caseworks corresponding to the size, orientation and shape of facial traits, which can be regarded as a soft biometric methods (Tome et al., 2015). The proposed features can also be deployed as additional information that can enhance the performance of traditional face attributes recognition. Experimental results indicate good recognition performance and high discrimination power especially for continuous features.

By comparing face attributes recognition algorithms to humans on challenging tasks, the level or accuracy of machines can be obtained. Both must recognize facial image taken under a variety of uncontrolled illumination conditions in both indoor and outdoor settings and also a person’s appearance that includes make up, wearing glasses and so on (Feng and Prabhakaran, 2016). In the good and middle condition, the algorithm performs far better than humans on face attributes recognition in pairing while in the poor condition, the humans result has more stable accuracy rather than algorithms (O'Toole et al., 2012). This research shows that the superiority of machines over humans in face recognitions field are not yet accomplished. A lot of study must be researched to build face attributes recognition for security, employment and so on.

1.2 Problem Statement

There are many fields of soft biometrics which apply the feature set selection and machine learning for recognition corresponding to experimental factors and fusion of soft biometric traits. It has not been focused yet in studying the application of soft biometrics in forensics (Nixon et al., 2015). There are many attributes in soft biometrics for face attributes recognition (Liu et al., 2015). Some attributes such as Asian, glasses and facial hair are not precisely recognized because the used dataset contains mostly the Caucasians with their own style and genetic. The lack of distinct dataset can lead to the decreasing performance of deep learning.

In order to produce efficient, robust and reliable face attributes recognition, it should undergo careful testing and verification on real-world datasets that are related to the real-world settings part. In the real-world, the face attributes recognition should be able to detect facial images in unavoidable face alterations, voluntary face alterations, uncontrolled environments and accuracy control in large-scaled dataset (Feng and Prabhakaran, 2016). The unavoidable face alteration factors include the aging impacts when identifying a face. The voluntary face alterations factors include camouflage, plastic surgery, make up and so on. The uncontrolled environments factors affect the accuracy of the algorithms especially on the small dataset which contains 264 augmented images (Romero Aquino et al., 2017). The increasing number of datasets can be used to improve accuracy neglecting the factors. In Figure 1.1, the large dataset is essential in recognizing many variability in a face. (Alza, 2017).

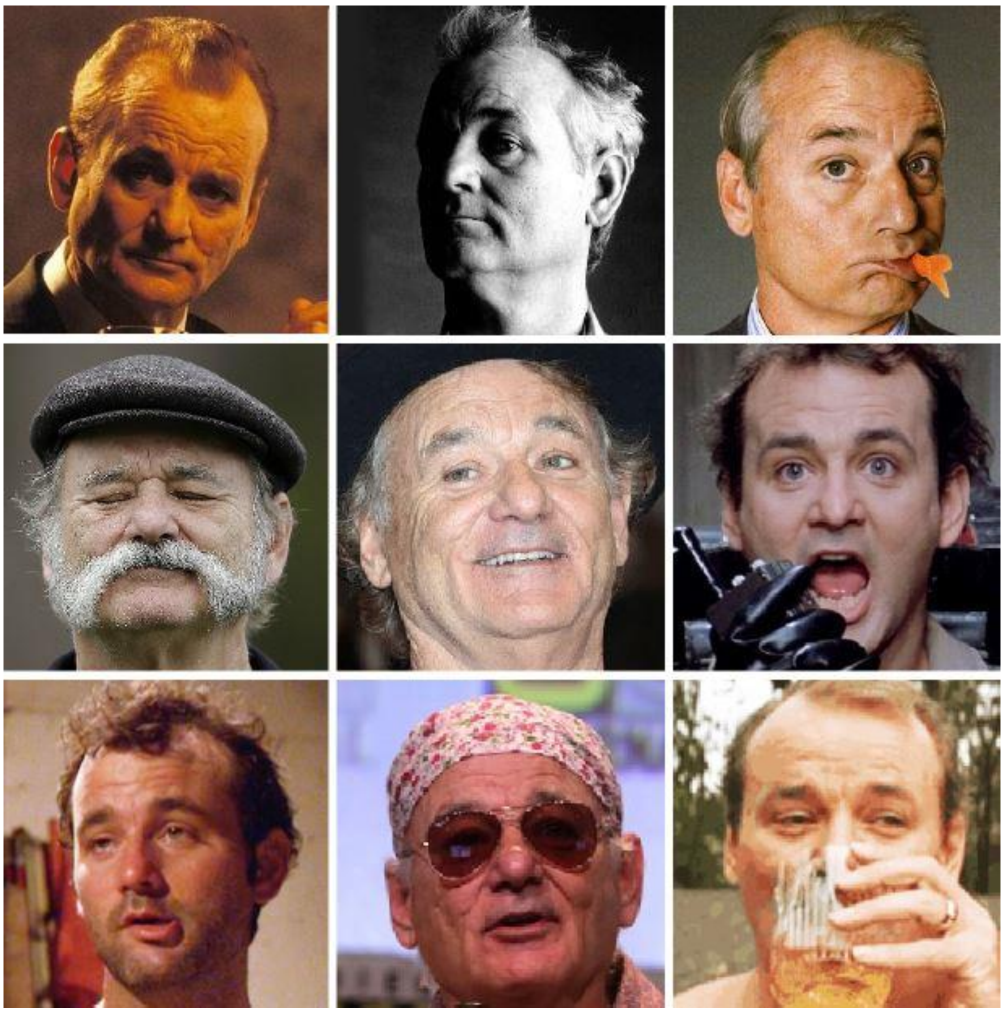


Figure 1.1: Variability in a face (Alza, 2017)

The face attributes recognition becomes more crucial in mobile phones application. In order to unlock the screen, the frontal facial image must be captured. Even this way of application is not considered as the most secure because the easy availability of frontal snapshots of the device owners from the social network, newspaper, magazines or other media (Findling and Mayrhofer, 2012). In Figure 1.2, the first two columns are examples of properly frontalized ones while in the last two there are cases in which frontalization has not performed properly (Alza, 2017). The convolutional neural network (CNN) failed many times in detecting the improper frontalized facial images.

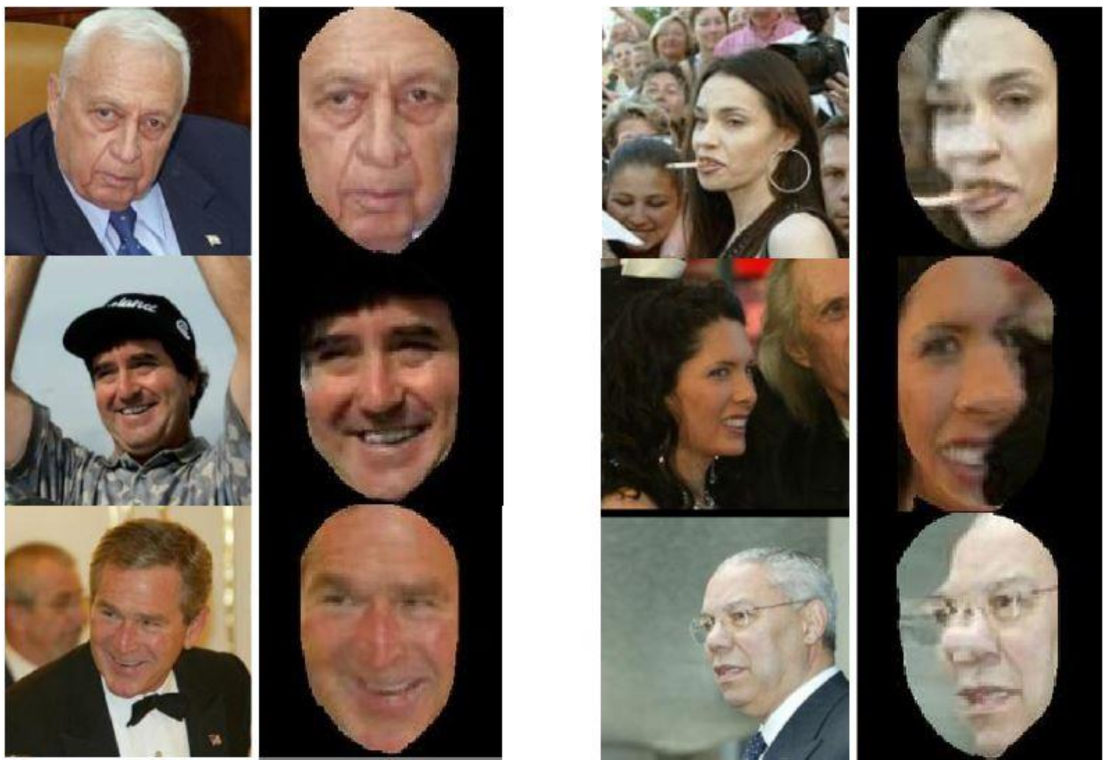


Figure 1.2: Examples of frontalized faces (Alza, 2017)

There are many ways of optimization for deep learning (DL). Hypothetically, increasing the convolutional layer increases the accuracy of performance. DeepID2+ shows an increment in accuracy with respect to the addition of convolutional layer (Sun et al., 2014) while Fast CNN shows the opposite results when 6 layer of convolution is added (Gopalakrishnan et al., 2017). For choosing optimizer, adaptive moment estimate (Adam) is experimentally proven in pavement distress detection (Gopalakrishnan et al., 2017) while nesterov-accelerated adaptive moment estimation (Nadam) is shown to be the best performance in training of word2vec word embeddings (Dozat, 2015). Both results are deployed in the application of face recognition since the research in this field commonly use stochastic gradient descent (SGD) as optimizer of DL (Luo et al., 2017).

1.3 Objective

The objectives of project are listed as follows:

1. To develop a system that recognizes the face attributes focusing on gender, glasses and facial hair.
2. To investigate the influence of dataset size on Fast CNN in gender attribute recognition.
3. To evaluate the gender attribute recognition system performance on improperly frontalized facial images.
4. To determine the best optimizer and configuration to train Fast CNN for gender attribute recognition.

1.4 Project Scope

The aim of this project is to build a training algorithm that is able to recognize face attributes as the soft-biometric information prior to facial recognition. It is noteworthy that facial recognition is not within the scope of this thesis. The Fast CNN is utilized as a feature extraction technique. Utilizing some optimizer in order to make the system more robust. The challenges involved in such a project is considered in order to obtain the predictable results.

Feature extraction is also a very challenging problem. Many algorithms and techniques have been utilized throughout the years such as Eigen Faces or Active Shape models (Çarıkçı and Özen, 2012, Edwards et al., 1998) . The recent application will produce the best results which is applying DL, particularly the CNN. Therefore, after studying the recent state of art, it has been concentrated in this project.

Finally, Python is used in implementing this project. Firstly, the face in the image is frontalized so that it is looking directly towards the camera. Next, the frontalized face is delivered to Fast CNN and a set of related features are extracted. Lastly, these features are applied as attributes to compare pairs of images to verify whether the sample are classified corresponding to gender, glasses and facial hair attributes.

CHAPTER 2

LITERATURE REVIEW

2.1 Overview

In this chapter, the flow of implementing project has been researched through a lot of paper, book and website. The introduction of this chapter has been explained in Section 2.2. The dataset which is the input of this project has been explained in Section 2.3. The library used in this work which is suitable for DL is explained in Section 2.4. A lot of model can be used for deep learning. Two models which are artificial neural network (ANN) and CNN are described in Section 2.5. The activation functions have been deployed in both model. These functions are explained in section 2.6. Section 2.7 explains the parameter that can be tuned in this project. Lastly, Section 2.8 explains the use of dropout in deep learning model.

2.2 Introduction

Artificial intelligence (AI) has been studied deeply till obtaining DL research (see Figure 2.1). AI, Machine Learning (ML) and DL have been described in Sections 2.2.1, 2.2.2 and 2.2.3 respectively.

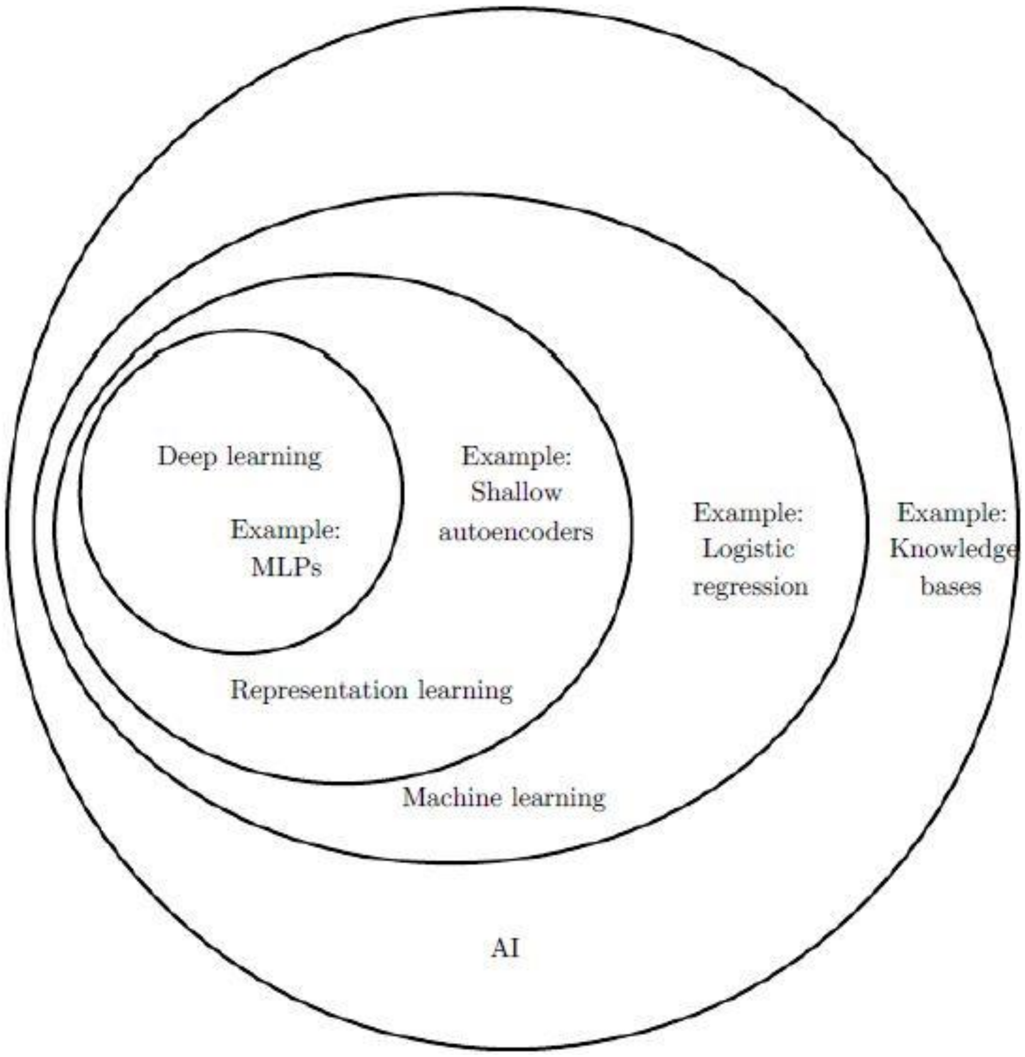


Figure 2.1: A Venn diagram of introduction (Goodfellow et al., 2016a).

In Figure 2.1, DL is a kind of representation learning (RL), which is in turn a kind of ML, which is applied for many but not all approaches to AI. Every section in the Venn diagram includes an example of an AI technology.

2.2.1 Artificial Intelligence

Nowadays, artificial intelligence (AI) is a successful discipline with numerous hands-on applications and active research areas (Goodfellow et al., 2016a). Intelligent software is considered to computerise every day work, recognise images or speech, perform diagnoses in medication and verify basic scientific study. During the previous days of AI, the discipline quickly handled and resolved issues that are mentally tough for humans but quite simple for computers—issues that can be explained by a list of formal, mathematical rules. The real task to artificial intelligence showed to be cracking the jobs that are simple for people to do but hard for people to explain formally on the issues that are solved naturally and automatically, like identifying spoken words or expression in images.

Several of the initial achievements of AI occurred in somewhat formal and sterile surroundings and did not need computers to have ample information about the world. For instance, IBM's Deep Blue chess-playing system triumph over world victor Garry Kasparov in 1997 (Hsu, 2002). Chess is certainly a very simple world, comprising merely sixty-four locales and thirty-two pieces that can shift only strictly limited ways. Planning an effective chess tactic is a huge achievement, but the challenge is not because of the complication of expressing the chess pieces set and permissible shifts to the computer. Chess can be totally expressed by a very short list of absolutely formal rules, simply kept ahead of time by the programmer. In contradiction, formal and abstract jobs that are among the toughest mental activities for a human are among the simplest for a computer. It is known that computers has the ability to beat even the greatest human chess player, but are only just matching not much of the capabilities of normal human beings to identify speech or objects. A human's daily life needs a huge amount of knowledge about the world. Many of this knowledge are subjective and instinctual, and hence hard to express formally. One of the main questions in AI is how to obtain this informal information into a computer.

Some AI plans have pursued hard-code information of the world in formal languages. A computer is capable of thinking about statements in these formal languages spontaneously using logical inference rules. This is called as the knowledge based method to AI. One of the most popular projects is Cycl (Elkan and Greiner, 1993). Cycl is an inference engine and a statements database in a language known as CycL. These statements are inserted by a human supervisor staff. It is a cumbersome procedure. People strive to create formal rules with sufficient difficulty to precisely explain the world.

2.2.2 Machine Learning

The complications confronted by systems depending on hard-coded information propose that AI systems require the capability to develop their own knowledge, by obtaining patterns from raw data. This capability is called machine learning. The overview of ML permitted computers to handle issues concerning real world knowledge and make choices that seem subjective. A simple ML algorithm known as logistic regression can decide whether to propose cesarean delivery (Mor-Yosef et al., 1990).

The execution of these simple ML algorithms relies greatly on the representation of the given data For instance, when logistic regression is utilised to suggest cesarean delivery, the AI system does not observe the patient directly (Goodfellow et al., 2016a). On the other hand, the doctor informs the system quite a few pieces of pertinent information, like the presence or absence of a uterine mark. Every chunk of information involved in the depiction of the patient is called as a feature. Logistic regression discovers how all of these features of the patient associates with numerous outcomes. Yet, it cannot affect how the features are described in any way. If an MRI scan of the patient was presented to the logistic regression, instead of the doctor's formalized report, it would not be capable of making useful estimations. Separate pixels in an MRI scan have insignificant connection to any difficulties that might happen during delivery.

An answer to this issue is to use ML to learn the mapping from representation to output as well as the representation itself. This method is called representation learning (Goodfellow et al., 2016a). Learned representations typically end in much better performance than can be gained with hand-designed representations. These also let AI systems to quickly adjust to new tasks, with minimum human involvement. The RL algorithm can learn a useful set of features for an easy task in short time or a difficult task in longer time. Designing the features manually for a difficult task needs a lot of human time and effort; it can take years for the whole community of researchers. The ideal example of a RL algorithm is the autoencoder. An autoencoder is the fusion of an encoder function that transforms the input data into another representation, and a decoder function that transforms the new representation back into the initial format (Goodfellow et al., 2016i). Autoencoders are trained to keep as much information as possible when an input is operated via the encoder and then the decoder, but are also trained to make the new representation have numerous fine properties. Different types of autoencoders target to attain different types of properties.

2.2.3 Deep Learning

DL lets the computer to create difficult concepts out of simpler concepts (Goodfellow et al., 2016a). Figure 2.2 depicts how a DL system can represent the idea of an image of an individual by merging simpler concepts, such as contours and corners, which are consecutively described in terms of edges. The ideal illustration of a DL model is the feedforward deep network or multilayer perceptron (MLP). A multilayer perceptron is simply a mathematical function mapping few set of input values to output values. The function is developed by combining many simpler functions. It can be considered that application of a different mathematical function as giving a new representation of the input. The notion of learning the correct representation for the data offers one viewpoint on deep learning. Another viewpoint on DL is that depth allows the computer to discover a multi-step computer program. Every layer of the representation can be considered as the state of the computer's memory after performing additional set of instructions in parallel. Networks with more depth can perform more instructions in order. Sequential instructions give big power because later instructions can refer back to the outcomes of previous instructions. In line with this perspective of deep learning, not all of the information in a layer's activations essentially encodes factors of variation that describe the input. The representation also keeps state information that aids to run a program that can add up of the input. This state information could be similar to a pointer or counter in an old-style computer program. It is not related with the content of the input particularly, but it assists the model to structure its processing.

The primary characteristic of DL is the capability of creating abstractions by building complex ideas from basic ones (Alza, 2017). Provided with an image, it is also capable of learning concepts such as cars, humans or cats by joining sets of simpler features such as edges and corners. This process is performed via successive "layers" that intensify the complexity of the learned concepts. The concept of depth in DL comes exactly from these abstraction levels. Each layer gains input and output of previous one and utilise it to learn higher-level characteristic, as shown in Figure 2.2. In few cases, the network is the one that utilise these characteristics to create an output, and occasionally it merely generates them from other techniques. If the layer increases, the abstract of concepts features represent also increases. This will continue until they are able to learn recognizing difficult concepts such as cars or people

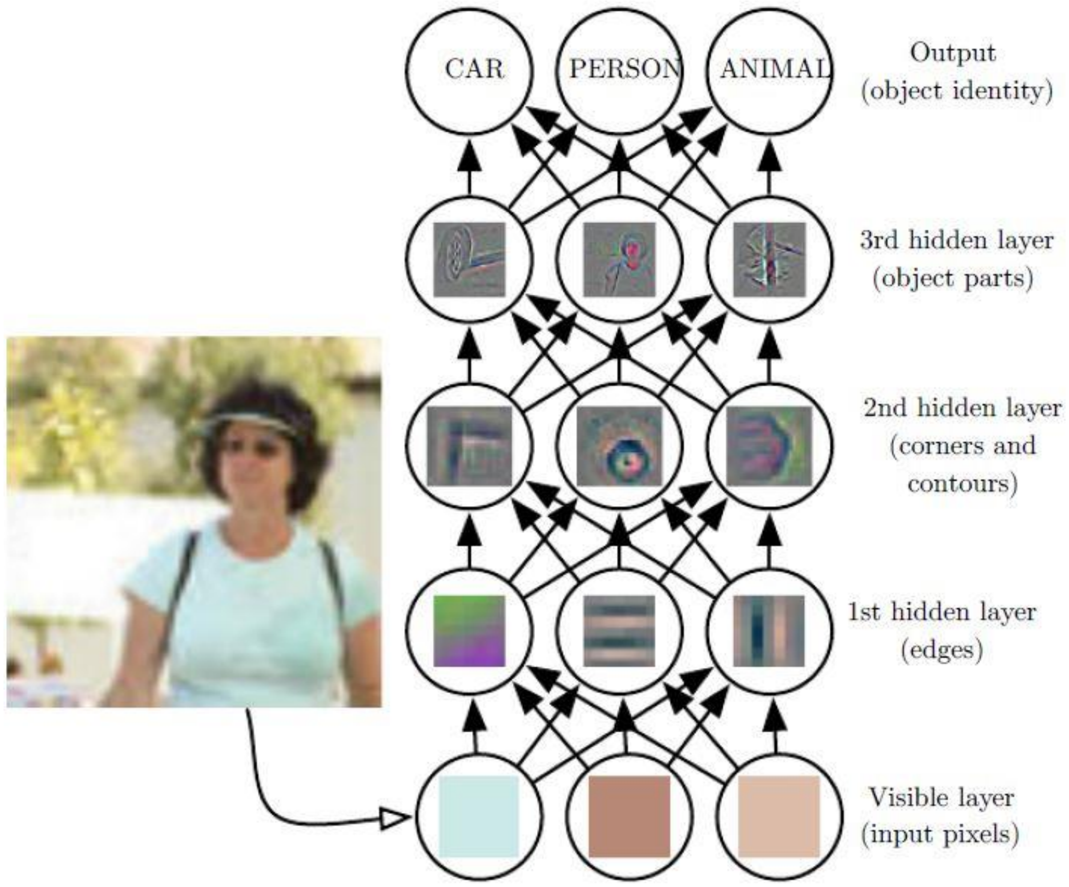


Figure 2.2: Illustration of DL (Goodfellow et al., 2016a).

2.3 Dataset

Soft biometrics in dataset is explained in the section 2.3.1 Dataset is the input of DL in this project in the form images which will be explained in the section 2.3.2. In Sections 2.3.3 and 2.3.4 explains the parameters deployed in dataset which are increasing size and augmenting dataset respectively.

2.3.1 Soft Biometrics

There are many research challenges in recognizing facial images. The challenges are unavoidable facial feature alterations, voluntary facial feature alterations, uncontrolled environments and accuracy on large-scale dataset. The unavoidable facial feature alterations includes the factors of age, health status, living habits and environments. These factors affect the facial feature changes such as pimple, wrinkle, cloudy eye and so on. For voluntary facial feature alterations, the pursuit of beauty causes all kinds of facial variations such as make up, hair style, dyed hair, contact lenses, glasses style and so on. For uncontrolled environments, the quality of media affects the accuracy of face recognition. For accuracy control on large-scale datasets, the extracted facial features algorithm must be scalable when applied to large-scale dataset (Feng and Prabhakaran, 2016). These challenges must be considered in order to get the best accuracy of facial images.

There are many ways for face attributes recognition. One of the ways is soft-biometric face attributes recognition. This way can detect the glasses and facial hair in facial images (Ouaret et al., 2010). In order to distinguish the gender of faces in the image, the features of faces can be extracted by using two methods which geometric features and appearance features. The first method applies an idea to detect the salient features and measure the distance between those salient features. The salient features include eyes, mouth and nose of the facial images. This method is also called as local features. The second method applies an idea to minimize a facial image by reducing the number of pixels. This method is also called as global features (Kumari et al., 2011). Therefore, male and female have a little bit difference based on these methods that will be used by DL system in classifying the gender.

In order to create high accuracy of classifying system, five factors of face aging internally and externally must be considered. The factors are gender, race, make up, health condition and ethnicity. These factors affect the aging pattern in the face image (Geng et al., 2006). The aging pattern must be computed with minimum reconstruction error. Therefore, CNN architecture is applied in order to get all the aging pattern for analysing the facial image (Aydogdu and Demirci, 2017). The CNN architecture will process the aging pattern in the facial image based on the number of pixels with respect to those factors. The facial biometrics had been applied on Mistral software (Charton et al., 2010).

2.3.2 Dataset Type

The data used with a convolutional network typically comprises of some channels, every channel being the different quantity observation at some point in time or space (Goodfellow et al., 2016f). Table 2.1 shows samples of data types with different dimensionalities and channel numbers. It has been discussed that only the case in which each example in the train and test data has identical spatial dimensions. One benefit to convolutional networks is that they can also manage inputs with different spatial degrees. These types of input basically cannot be exemplified by traditional, matrix multiplication-based neural networks. This gives a convincing reason to use convolutional networks even when computational cost and overfitting are not major issues. For instance, take a group of images, in which every image has a different height and width. It is vague how to model such inputs with a weight matrix of permanent size. Notice that the application of convolution for dealing with variable sized inputs only sensible for inputs that have variable size because different recordings lengths over time and different observations widths over space.

Table 2.1: Examples of different formats of data that can be employed with convolutional networks.

Number of dimensional array	**Single channel**	**Multi-channel**
1-D	Audio waveform: The axis is convolved over corresponding time. Time is discretized and measured the amplitude of the waveform once per time step.	Skeleton animation data. Animations of 3-D computer-rendered characters are produced by changing the pose of a "skeleton" over time. At every point in time, the pose of the character is described by a specification of the angles of every joints in the character's skeleton. Every channel in the data is fed to the convolutional model represents the angle about one axis of one joint.
2-D	Audio data that has been pre-processed with a Fourier transform. The audio waveform is transformed into a 2D tensor with different rows corresponding to different frequencies and columns corresponding to different points in time. Using convolution in the time	Colour image data. One channel has red, green and blue pixel. The convolution kernel moves over both the horizontal and vertical axes of the image, conferring translation equivariance in both directions.

	makes the model equivariant to shifts in time. Using convolution across the frequency axis makes the model equivariant to frequency, so that the same melody played in a different octave generates the similar representation but at a different height in the network’s output.	
3-D	Volumetric data. A common source of this kind of data is medical imaging technology, such as CT scans.	Colour video data. One axis corresponds to time, one to the height of the video frame, and one to the width of the video frame.

2.3.3 The Increasing Number of Dataset

The main thoughts are the feasibility and cost of collecting more data, the feasibility and cost of decreasing the test error by other means, and the quantity of data that is estimated to be compulsory to enhance test set performance considerably (Goodfellow et al., 2016g). At huge internet corporations with millions or billions of users, it is possible to collect large datasets, and the cost of doing so can be significantly less than the other options, so the solution is practically always to collect more training data. For instance, the improvement of huge labelled datasets was one of the most vital reasons in resolving object recognition. In other circumstances, such as medical applications, it may be expensive or impossible to collect more data. A simple option to collecting more data is to decrease the model size or enhance regularization, by modifying hyperparameters such as weight decay coefficients, or by including regularization approaches such as dropout. Collecting more data is appropriate if the gap between train and test performance is still deplorable even after tuning the regularization hyperparameters. If collecting much more data is not achievable, another method to enhance generalization error is to develop the learning algorithm itself. This becomes the field of research and not the field of advice for applied experts.

2.3.4 Dataset Augmentation

For several ML jobs, it is rationally clear-cut to generate new fake data (Goodfellow et al., 2016d). This method is the simplest method for classification. A complex, high dimensional input *x* should be taken by a classifier and summarized with a category identity *y*. This denotes that the major task coping with a classifier is to be invariable to a large range of transformations. New (*x, y*) pairs can be produced effortlessly just by converting the *x* inputs in our training set. This method is not as freely relevant to many other tasks. For instance, it is quite hard to produce new fake data for a density estimation task except if the density estimation problem managed to be solved. Dataset augmentation has been a specifically efficient method for a particular classification problem which is object recognition. Images are high dimensional and involve a wide variety of variation factors, many of which can be simply faked. Processes like transforming the training images a few pixels in every direction can usually significantly enhance generalization, although the model has already been invented to be partly translation invariant by using the convolution and pooling methods explained in Sections 2.11 and 2.12 respectively. Several other processes like rotating the image or scaling the image have also verified to be quite efficient.

The models are trained with different approaches using data augmentation to enhance their classification competence (Romero Aquino et al., 2017). The models are also trained and verified without any augmentation in order to contemplate those results as initial baselines. Random transformations were used on the raw images of the train dataset to augment them (cropping, rotation, swirl, horizontal flip, vertical flip and different forms of noises such as salt and pepper, and Gaussian noise). The test set remained unaffected in all cases.

Traditional transformations (Perez and Wang, 2017) use a combination of affine transformations to influence the training data (Jannik Bjerrum, 2017). For every input image, a ”duplicate” image is generated, which is shifted, rotated, zoomed in/out, distorted, flipped, or shaded with a hue. Both image and the duplicate are placed into the neural net. For a dataset of size N, a dataset of 2N size is generated.

Neural augmentation (Perez and Wang, 2017) functions outstandingly better than no augmentation. In the dogs versus cats problem, the neural augmentation works the best with a 91.5% to 85.5% in comparison with no augmentation. In the dogs verses fish problem, the neural augmentation works the second best with 77.0% to 70.5%. Whereas the traditional augmentation works almost as well at a lesser time expense, this does not prevent the combination of different augmentation approaches. An approach that first operates traditional augmentations, and then pairs up data for neural augmentation, which could possibly beat all experiments tested.

2.4 Library for Deep Learning

DL is an continuously-growing ML methods based on data representations (Clark, 2018). The vast amount of resources can be a touch overwhelming for those either looking to get into the field, or those already engraved in it. A good way of staying updated with the latest trends is to interact with the community by interacting with the DL open source projects that are currently available. Based on Figure 2.3, Tensorflow is the best DL library followed by Keras.

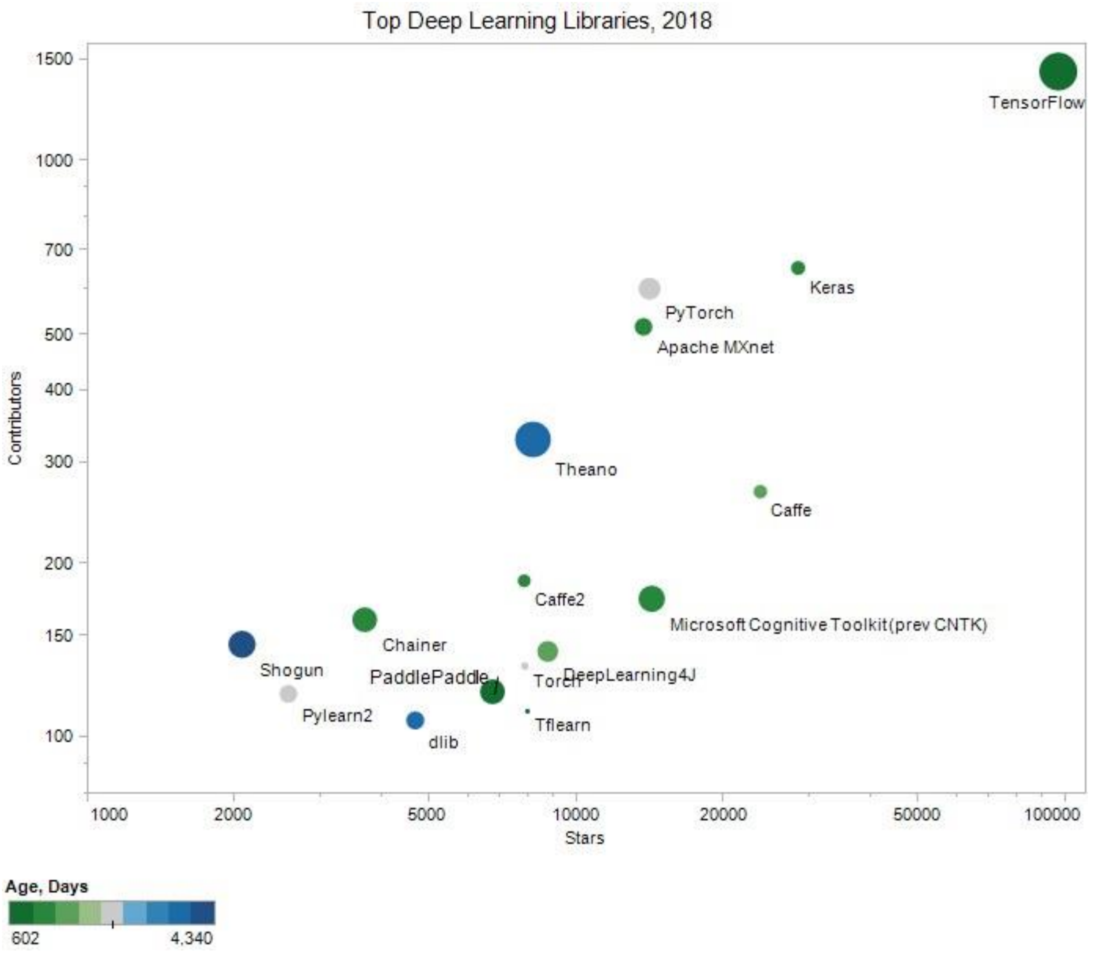


Figure 2.3: Top 16 open source DL libraries by Github stars and contributors (Clark, 2018)

2.4.1 Tensorflow

TensorFlow signifies calculations by dataflow graphs (Mart et al., 2017). While focusing on ML applications, TensorFlow is quite sceptical on the exact idea of the calculations. To be specific, a computation may execute one or more training steps for a machine-learning model, or it can be the application of a trained model. This indicates that dataflow graphs support both inference and training. A dataflow graph is comprised of nodes and edges, in which every node symbolises an instantiation of an operation, and values flow by the edges. The operations are executed by kernels that can be operated on certain types of devices.

The key interest values are tensors, which are arbitrary dimensionality arrays where the underlying element type is identified or inferred at graph-construction time (Mart et al., 2017). For that reason, the operations are mostly consisting of mathematical functions like matrix multiplication. Furthermore, certain operations may update or read state. In TensorFlow, a variable is an exceptional type of operation that returns a handle to a tensor. In this situation, informally, it can be said that the tensor is held in the variable, in which the variable operation and the resulting handle may be conflated. The handle can be approved as argument to operations that update or read the equivalent tensor. To illustrate, the tensor may consist of the weights of a layer in a neural network, and these are updated through the training process. Besides, along with the edges for communicating tensors, a graph may incorporate control edges that restrict the order of execution. In the existence of mutable state, this order can affect the observable semantics, which as well affect performance.

2.4.2 Keras

Following best practices to reduce cognitive load, Keras (Chollet, 2015a) provides consistent & simple API, reduces the number of user actions needed for common use cases, and offers clear and actionable feedback on user error. A model is seen as a series or a graph of standalone, fully-configurable modules that can be plugged together using the least possible restrictions. Specifically, optimizers, cost functions, regularization schemes, initialization schemes, neural layers, and activation functions are all standalone modules that can be combined to invent new models. New modules are easy to add as new classes and functions. The current modules give plenty examples. The ability to easily design new modules allows for total expressiveness, making Keras appropriate for advanced research. No distinct models configuration files in a declarative format. Models are explained in Python code, which is compact, easier to debug, and allows for ease of extensibility.

2.5 Deep Learning Model

Several models such as artificial neural network (ANN), CNN, recurrent neural network (RNN) and deep bolltzman machine (DBM) can be deployed in DL. Since the data input is in the form of images, ANN and CNN is suitable for this project. Section 2.5.1 explains ANN while section 2.5.2 explains CNN.

2.5.1 Artificial Neural Network

ANN derives from a group of computationally simple nodes that are connected together called neurons (Alza, 2017). The neurons are arranged in layers linked between them, just like the biological neurons are linked with axons. These layers are distributed into three main types which are input, hidden and output. The input layer parallels to the data received by the network. It could be comprehended as the input vector from other techniques. This layer is linked to a hidden layer, the one that is not in extreme. This is how it got its name as it is “invisible” from the outside. Another interesting explanation which is different to other approaches is, once the network is trained, it does not give any insight on what it does. The ANN sometimes called as black boxes, as it is almost impossible to comprehend its functions. It can be several hidden layers, each linked to the previous one. Each neuron in hidden and output layers is usually linked to all neurons from the previous layer. Every edge possesses an associate weight, which shows how strong the two neurons are linked, either directly or inversely like how the biological neurons are linked. Lastly, the last layer, known as output layer, provides the result of the ANN with an output for each class. This is crucial because ANN is typically utilized for classification problems.

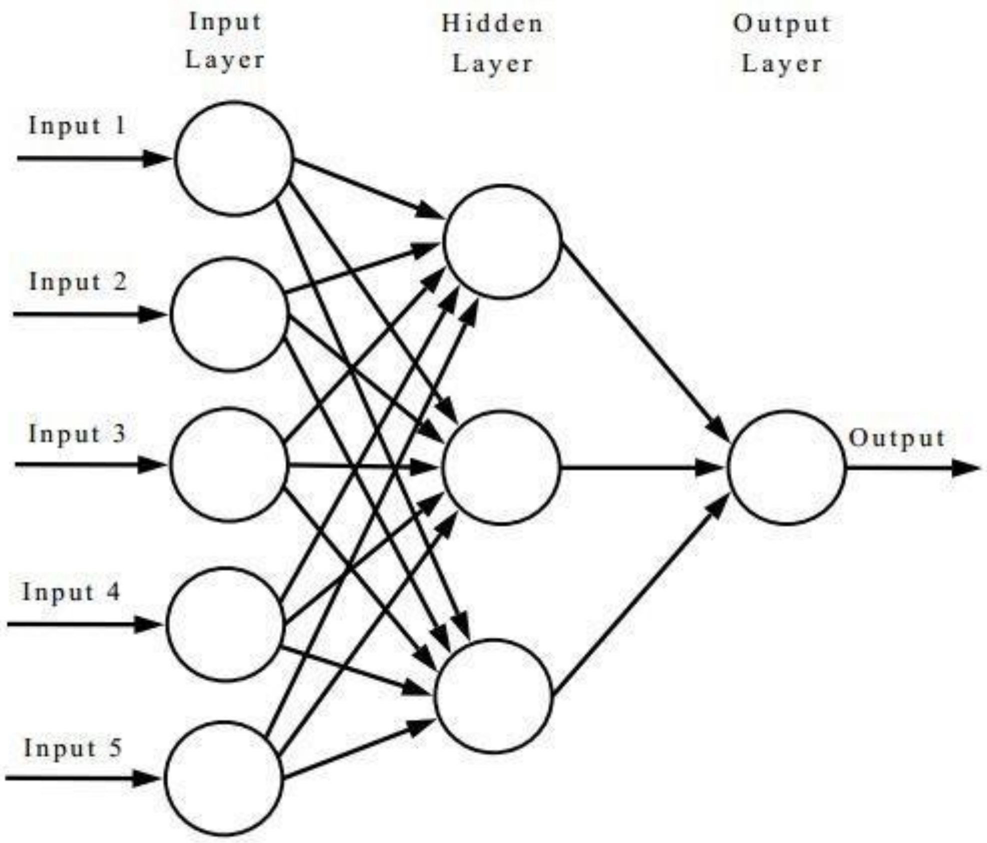


Figure 2.4: The three layers of an ANN (Shi, 2014).

In Figure 2.4, ANN is used to estimate an unknown mathematical function. The function can be either linear or non-linear. Theoretically, ANN has the ability to estimate any function. The neuron, its basic unit, calculates a “simple” activation function given its inputs, and spread its value to the subsequent layer. Thus, the entire function is formulated by collecting activation values from all neurons. Having hundreds of neurons, the number of edges can reach higher orders of magnitude, and also the difficulty in interpreting them.

2.5.2 Convolutional Neural Network

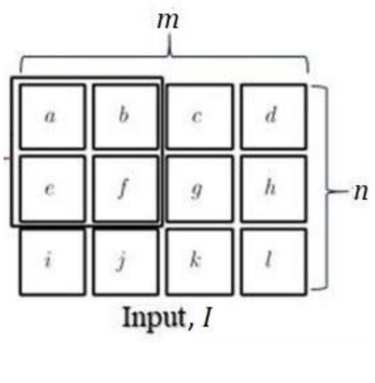
The most commonly applied deep neural network (DNN) in CV problems are the CNN, which is based in the MLP architecture (Alza, 2017). ANN on the other hand, are inspired in general neuronal behaviour, and CNN follow the same rules as animal’s visual cortex. This contains neurons that process only small segments of the input image, or visual field, and recognizes related patterns. These neurons are compacted in structures like layers that allows intensify complex patterns. It is almost like the general DNN structure, but it is different in terms of shared weights

2.5.2.1 Convolutional Layer

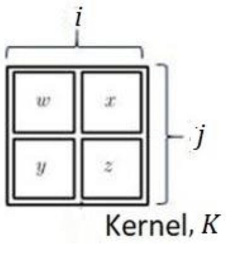
In ML exercises, the input is normally a multidimensional array of data and the kernel is normally a multidimensional array of parameters modified by the learning algorithm (Goodfellow et al., 2016f). These multidimensional arrays are referred as tensors. Since every component of the input and kernel must be clearly stored individually, it is commonly assumed that these functions are nil everywhere but the limited set of points for which the values are stored. This indicates that in fact, the infinite summation can be implemented as a summation over a limited number of array elements. Lastly, convolutions are frequently used over more than one axis at a time. For instance, if a two-dimensional image I is used as an input, a two-dimensional kernel K will probably be used as well.

$$S(i,j) = (I * K)(i,j) = \sum_{m} \sum_{n} I(m,n)K(i-m, j-n) \tag{2.1}$$

As shown in Figure 2.5, $m \times n$ is the matrix of input, I while $i \times j$ is the matrix of kernel, K. The mapping of kernel matrix on input matrix is indicated by $K(i-m, j-n)$.



(a) Input, I matrix



(b) Kernel, K Matrix

Figure 2.5: The illustration of input, I and kernel, K matrices

Since convolution is commutative, it can be equivalently expressed as:

$$S(i,j) = (K * I)(i,j) = \sum_{m}\sum_{n} I(i-m, j-n)K(m,n)$$

(2.2)

Generally the second formula is more clear-cut to apply in the ML library, as there is less difference in the range of valid values of m and n. The commutative convolution property occurs because the kernel relative has been flipped to the input. If m increases, the index into the input also increases, while the index into the kernel lessens. The only purpose of flipping the kernel is to get the commutative property. Although the commutative property is helpful in writing proofs, it is not normally a vital property of a neural network application. On the other hand, numerous neural network libraries execute a correlated function named the cross-correlation, which is similar to convolution but without flipping the kernel:

$$S(i,j) = (I * K)(i,j) = \sum_{m}.\sum_{n} I(i+m, j+n)K(m,n)$$

(2.3)

A lot of ML libraries apply cross-correlation using the term convolution (Goodfellow et al., 2016f). This section will stick to the practice of calling both operations as convolution, and stipulate whether the kernel is meant to be flipped or not in the situation where kernel flipping is appropriate. In the ML, the learning algorithm will learn the suitable kernel values in the proper place, thus an algorithm based on convolution with kernel flipping will discover a flipped kernel in relation to the kernel discovered by an algorithm without the flipping. It is also uncommon if convolution is utilised alone in machine learning; instead of using it concurrently with other functions, and the grouping of these functions does not commute irrespective of whether the convolution operation flips the kernel or not. Figure 2.6 is an illustration of convolution utilised to a 2-D tensor.

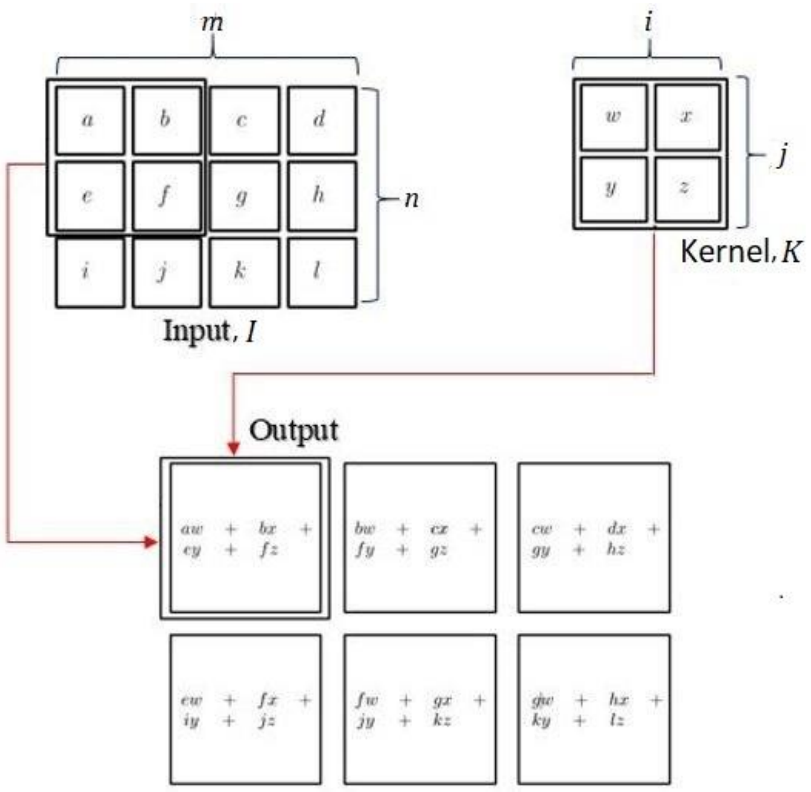


Figure 2.6: Example of 2-D convolution without kernel-flipping (Goodfellow et al., 2016f).

In Figure 2.6, the output to only positions is restricted where the kernel lies entirely within the image, known as "valid" convolution in some contexts. The boxes are drawn with arrows to show how the upper-left element of the output tensor is formed by applying the kernel to the corresponding upper-left region of the input tensor.

2.5.2.2 Pooling Layer

A pooling function substitutes the output of the net at a certain location with a summary statistic of the nearby outputs. As an example, the max pooling (Zhou and Chellappa, 1988) operation reports the maximum output within a rectangular neighbourhood. Other functions are including the average of a rectangular neighbourhood, the L_2 norm of a rectangular neighbourhood, or a weighted average based on the distance from the central pixel.

In Figure 2.7, pooling makes the representation become approximately invariant to small translations of the input (Goodfellow et al., 2016f). Invariance to local translation can be a very useful property in order to locate some important feature. The use of pooling can be considered