CZECH TECHNICAL UNIVERSITY IN PRAGUE
FACULTY OF MECHANICAL ENGINEERING
DEPARTMENT OF INSTRUMENTATION & CONTROL

Image Recognition with Deep Learning for Web Scrapped Images

BACHELOR THESIS

by

Ahmed Toubar

Supervisor: Doc. Ing. Ivo Bukovský, Ph.D

Annotation Sheet

Name	**Ahmed**
Surname	**Toubar**
Title Czech	**Rozpoznávání snímků s hlubokým učením pro snímání webu**
Title English	**Image Recognition with Deep Learning for Web Scrapped Images**
Academic year	**2016/2017**
Language	**English**
Department	**Instrumentation & Control**
Specialization	**Information & Automation Technology**
Supervisor	**Doc. Ing. Ivo Bukovský, Ph.D.**
Reviewer	**Ing. Vladimír Hlaváč, Ph.D.**
Keywords	**Deep Learning; Artificial Intelligence, Python, TensorFlow, Computer Vision; Object Recognition in Images**

FACULTY OF MECHANICAL ENGINEERING CTU IN PRAGUE

BACHELOR THESIS ASSIGNMENT

I. PERSONAL AND STUDY INFORMATION

Name:	**Essameldin Toubar Ahmed Mohamed Alaa**	Personal no.: **456145**
Faculty:	**Faculty of Mechanical Engineering**	
	Department of Instrumentation and Control Engineering	
Study program:	**Engineering**	
Study field:	**Information and Automation Technology**	

II. BACHELOR THESIS SPECIFICATION

Bachelor thesis title (in English):

Image Recognition with Deep Learning for Web Scrapped Images

Bachelor thesis title (in Czech):

Rozpoznávání obrazů s technikou deep learning pro obrázky z webu

Elaboration guidelines:

1) Review the principle of deep learning for object recognition in images (ORI).
2) Review the computational platforms for deep ORI and select one for your purpose and technically reason for your selection (e.g. by comparison to others).
3) Collect a database of specific oriented images (>1000) displaying certain type of objects that can be possibly at various
coordinates, with possible varying rotation, of various object size, background etc...
4) Apply deep learning for ORI so the deep learned model (neural network) returns a limited number of pictures (<10) with
the most similar objects (explain how you define similarity for your purpose))
5) Investigate the performance of deep learning for ORI and document your work with proper technical details.

Literature resources:

[1] SCHMIDHUBER, Jürgen. Deep learning in neural networks: An overview. Neural Networks [online].
[2] NIELSEN, Michael A. Neural Networks and Deep Learning [online]. 2015
[3] Neural networks and deep learning [online]. (http://neuralnetworksanddeeplearning.com/)

Name and affiliation of the thesis supervisor:

doc. Ing. Ivo Bukovský Ph.D., Dpt. of Instrumentation and Control Engineering, FME

Consultant:

-

Assignment issue date: **19th April 2017** Thesis due date: **16th June 2017**

Validity of assignment: -

Thesis Supervisor | Head of the Department | Dean of the Faculty

III. THESIS ASSIGNMENT RECEIVED

The student is aware that the thesis has to be accomplished through an independent and unassisted student's work, supported only by recognized consultations. Literature and other information resources as well as consultants' names have to be acknowledged in the thesis.

19.04.2017

Date

Ahmed Toubar

Student

Declaration

I declare that this bachelor thesis entitled “Image Recognition with Deep Learning for Web Scrapped Images” is my own work performed under the supervision of Doc. Ing. Ivo Bukovský, Ph.D with the use of the literature presented at the end of my bachelor thesis in the list of references.

In Prague 15.06.2017 Ahmed Toubar

Signature

Acknowledgments

I would like to express my special thanks to my supervisor Doc. Ing. Ivo Bukovský, Ph.D for his continuous devotion, invaluable support and intellectual guidance through the course of my work.

Abstract

This thesis aims to explore Object Recognition in Images (ORI) using Artificial Intelligence (AI) in the general sense and Deep Learning in the more specific sense. AI has gone through a lot of phases ever since the term was first coined by the computer scientist John McCarthy in 1955[1]. In this paper, I will first loot at Web Scraping of data to put my hands on a worthy data set of images which I will use to apply ORI using AI. Then, I will explore the evolution of AI to grasp a general sense of how it is transforming the world that we live in today. After that, I will investigate and analyse the different computational platforms that are currently available for users to harness the power of AI. Finally, I will evaluate the efficiency and effectiveness of AI as a means of ORI.

Table of contents

List of tables

List of figures

1 Introduction

The future that once was imagined in science fiction movies is now a part of our reality. It is undeniably true that we, humans, are on the verge of a revolution that will change our lives forever – The Artificial Intelligence (AI) Revolution[1].

It all started with the industrial revolution in the 1800s that changed how we interact with the world in every possible way. It has disrupted transportation, communication and raised the average standard of living by offering jobs to workers at factories that never existed before[2]. But still there were a lot of manual wearing tasks that consumed one's mind, effort and time. The valuable resources that lets a person be creative and innovate relentlessly.

Then, in 1926, Julius Edgar Lilienfeld patented the transistor. The single component that opened the gate to yet another revolution - The Computer Revolution. The invention of the computer again changed how we interact with the world and with each other. It has put an end to manual exhausting tasks and became the gateway to limitless innovation[3]. However, those computers were distant and did not communicate effectively. And so, came the Internet.

Meanwhile, two remarkable gentlemen were working on a communication system experiment to replace their existing networking system at ARPANET. This experiment went out of hands and laid the foundations of what we know today as the internet[4]. The internet came to revolutionize informatics and communication. It disrupted every industry there is, created jobs, changed the world to the better and more importantly left us with enormous amounts of data. Data that can make computers learn like us humans do.

Until now in our timeline, computers were programmed to do every single task. A programmer had to explicitly instruct it, for example, to store the value of X and Y, then add them together and print the result to the screen. This was still

one of our greatest inventions as humans. But it made computer scientists think about the what if – What if computers can think for themselves?
And so, John McCarthy thought about this particular question and coined the term “Artificial Intelligence” in 1955 at the world’s first AI conference. He was the pioneer that imagined the possibility of computers reasoning like how humans do. That paved the way to what is currently taking place in our world – An AI Revolution[5]. In the upcoming chapter I will discuss the evolution of AI and how it is transforming how we interact with the world today.

1.1 Artificial Intelligence (AI)

Artificial Intelligence (AI) is defined as the intelligence executed by machines that take actions driven towards succeeding at a goal. It leverages self-learning systems and uses various tools such as pattern recognition and data mining. It functions in a way similar to how a normal human brain works in day-to-day tasks from common-sense reasoning to behaving in a social manner. AI has been one of science’s most incomprehensible subjects in Computer Science due to its significance in our everyday lives and the massive potential it has to continue changing people’s lives. It is considered to be the fourth industrial revolution where automation will be intense and connectivity will be very abundant. The applications that AI has are vast and are in the way computers are being used in society.[6], [7]

1.2 History & Evolution

Although the term Artificial Intelligence was first used at a conference on the subject in 1956 by John McCarthy, the journey to understanding if machines can think began even before that. A system which showcases people’s own knowledge and understanding of artificial intelligence was proposed by Vannevar Bush, and only 5 years later, a paper was written by Alan Turing discussing the topic of machines being able to simulate human beings and

perform intelligent activities such as playing Chess. In his landmark paper, Turing noted that the term “thinking” is difficult to define and derived his famous Turing Test, which states it was reasonable to say that a machine is thinking that if it could carry out a conversation via a teleprinter that could be identified as indistinguishable compared to a conversation with a human being. Turing Test was utilized as a green light to argue that a thinking machine was at least possible and it was the first serious proposal in the study of artificial intelligence. [7]

There have been numerous advances over the past 60 years mainly in search and machine learning algorithms and integrating statistical analysis. Artificial Intelligence is divided broadly into three phases: artificial narrow intelligence (ANI), artificial general intelligence (AGI) and artificial super intelligence (ASI). The first stage is restricted to only focusing on one functional area. AGI, the second stage, covers multiple fields such as abstract thinking, reasoning and problem solving and is considered an advanced level. The final stage (ASI) is where artificial intelligences outperforms and exceeds human intelligence throughout all fields. [6]

As displayed in the graph below, the transition between ANI and AGI (first and second phases) has taken a long time. Scientists believe that we are nowadays almost completing ANI, the first phase, and will slowly be completing the transition to the second phase, AGI, where the intelligence of machines is being equated to that of humans, which is a drastic achievement. It is believed that by the end of the decade, we will be entering the phase AGI and the industry of artificial intelligence is expected to start exploding and its impact on society and businesses will begin emerging.[6]

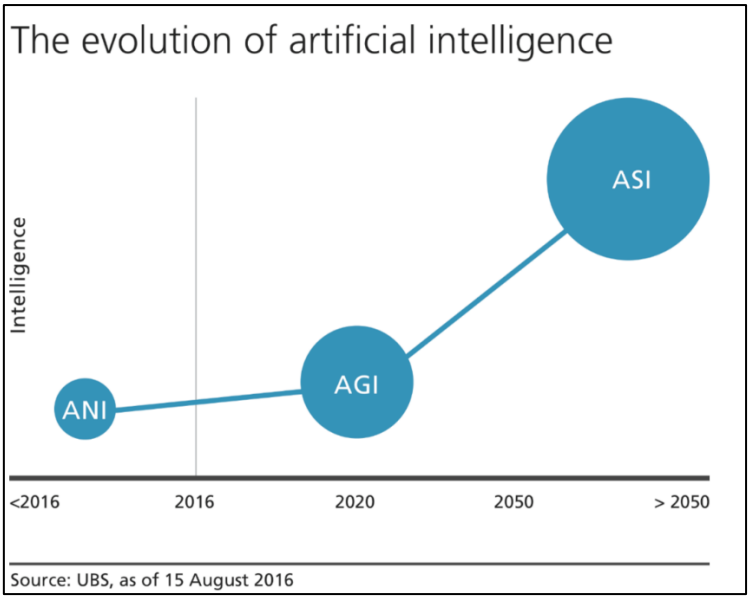


Figure 1-1 – Evolution of AI

By the end of the decade, a true autonomy will commence. AI-powered software and machines will no longer need human supervision and will embark on a new path of them becoming alert beings. Having said that, the next few years will witness tremendous industry growth and will have a remarkable expansion on its expansion on businesses and society. Estimates show that by 2020, the industry's marketplace by revenue should more than double compared to it's value in 2015. As shown in the graph below, at a 20% annual growth rate, it is expected to become a USD 12.5 billion industry driven by exponential improvements. [6]

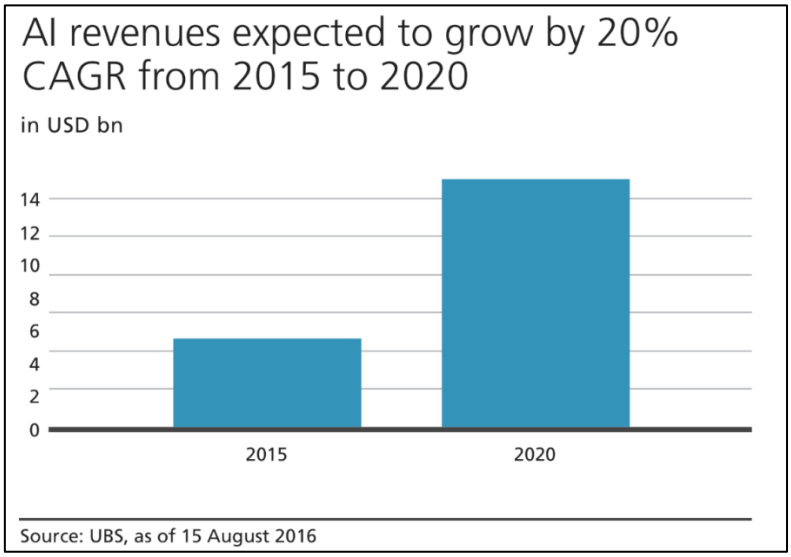


Figure 1-2 – AI Revenue growth expectations

1.3 How did Data become so Valuable?

Our world today hosts an unbelievably remarkable vast amount of data in digital format and it's getting even vaster very quickly. This amount of digital information makes it possible to do things that could not be done in the past such as prevent diseases, defeat crime and identify business trends. An example includes Wal-Mart, an American retail giant, which handles more than a million customer interactions per hour and feeds "databases estimated at more than 2.5 petabytes—the equivalent of 167 times the books in America's Library of Congress" [8]

The unimaginably vast amount of data available in today's world can be utilized to unlock new sources of monetary and economic value as well as provide new insights into science if managed well. The biggest and most obvious reason for the information explosion is technology. The functions and capabilities of digital devices are massively increasing and their prices are plummeting, which means that sensors are digitising massive amounts of information that was historically unavailable. There are currently more than 4.5 billion mobile phone subscriptions in the world and almost 2 billion people use the internet, which opens up endless doors for data collection of all sorts. [8]

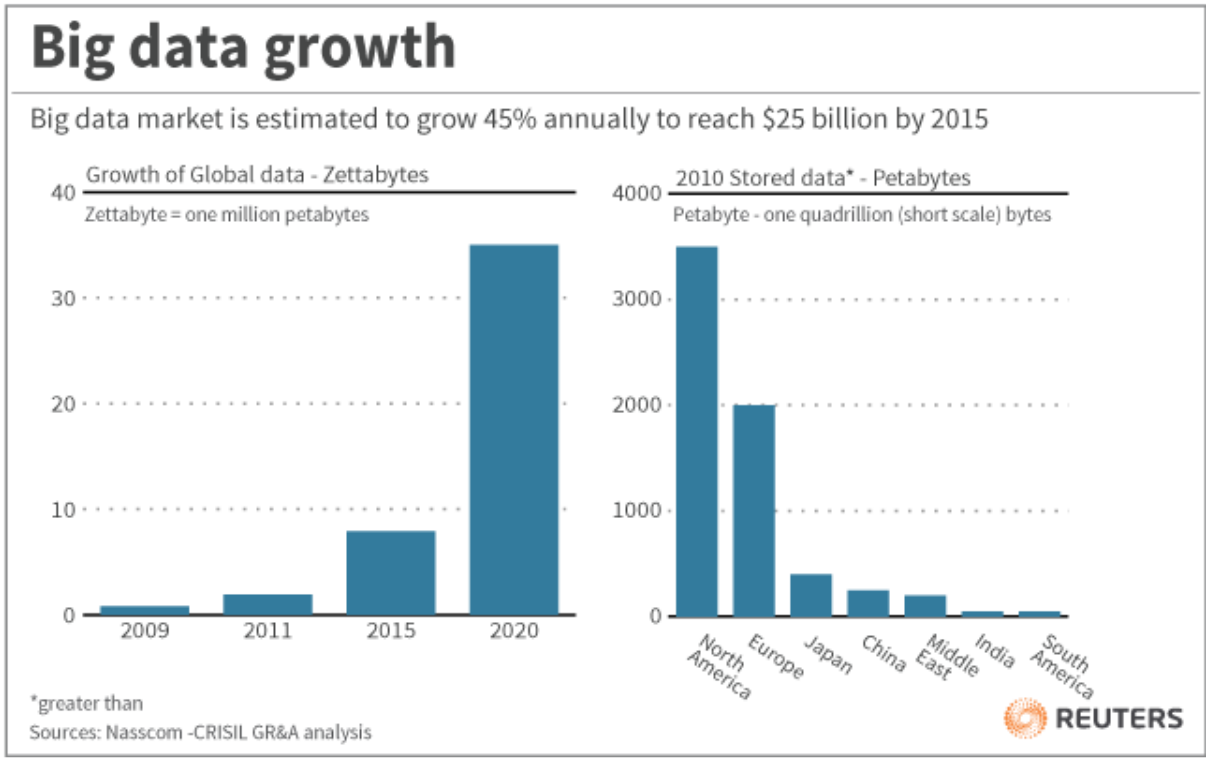


Figure 1-3- Global Data Growth

As we can see from figure 1-3, there is a global exponential increase in data growth year over year. This has led to the need of innovative brand new methods for data processing and storage. And so, the machine learning and then deep learning fields emerged.

2 Deep Learning

2.1 Introduction

2.1.1 What is Deep Learning?

An aspect of machine learning that emerged from the overlapping of signal processing, optimization, neural networks, pattern recognition and artificial intelligence is now referred to as deep learning. Models of deep learning have been honored by exemplary scholars in renowned academic journals. Businesses of all sizes are now able to utilize deep learning to revolutionize real-time data analysis thanks to cheaper memory and faster computer processors. [9]

Deep learning can consist of either supervised or unsupervised learning from data with the use of several machine learning aspects. These layers constitute of multiple stages of nonlinear transformations where the characteristics of data are being represented at a higher and more complex layers. The two major basic types used in data science are **supervised learning** and **unsupervised learning**. In supervised learning, the training data contains known outcomes and the model is trained with relevance to those specific outcomes. On the other hand, in unsupervised learning, the training data does not contain any known outcomes and the algorithm self-discovers and identifies patterns and relationships within the data. [9]

Figure 2-1 is referenced when it comes to all types of machine learning models developed. Input Data is passed through the model and is filtered out across multiple non-linear layers. The final layer of the model constitutes of a simple classifier that determines the class to which the object falls under.

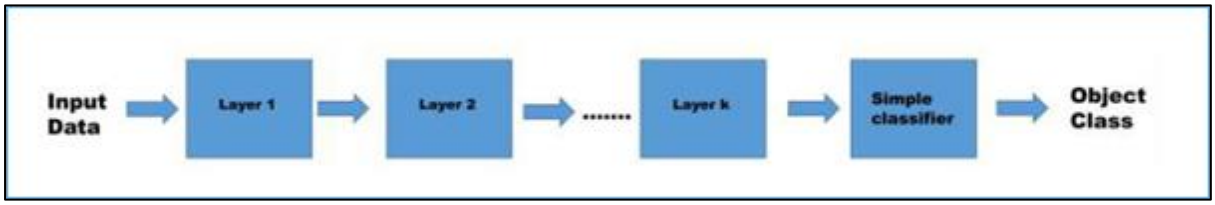


Figure 2-1- General deep learning framework

Predicting a response that is variable using given attributes is the goal of learning from data, which is quite similar to linear regression, where a linear model is used to predict a dependent variable (response) using several independent variables. Having said that, however, traditional linear regression is not considered deep as multiple layers of nonlinear transformation are not being applied with the data. In addition, data techniques such as decision trees, support vector machines and random forests are also not considered deep, although they are powerful tools. Random forests and decision trees utilize original input data with no transformations generated, same as with support vector machines which only consist of linear transformation, hence considered shallow. In addition, single hidden layer neural networks are considered shallow because they solely consist of one hidden layer. [9]

2.1.2 What Problems Can Deep Learning Solve?

Deep learning models are unique in the sense that they are able to classify or predict nonlinear data using a number of parallel nonlinear steps. Deep learning models learn the features of input data hierarchy starting from raw data input all the way to the actual classification of data. Each layer uses the extract features of the output from previous layers.

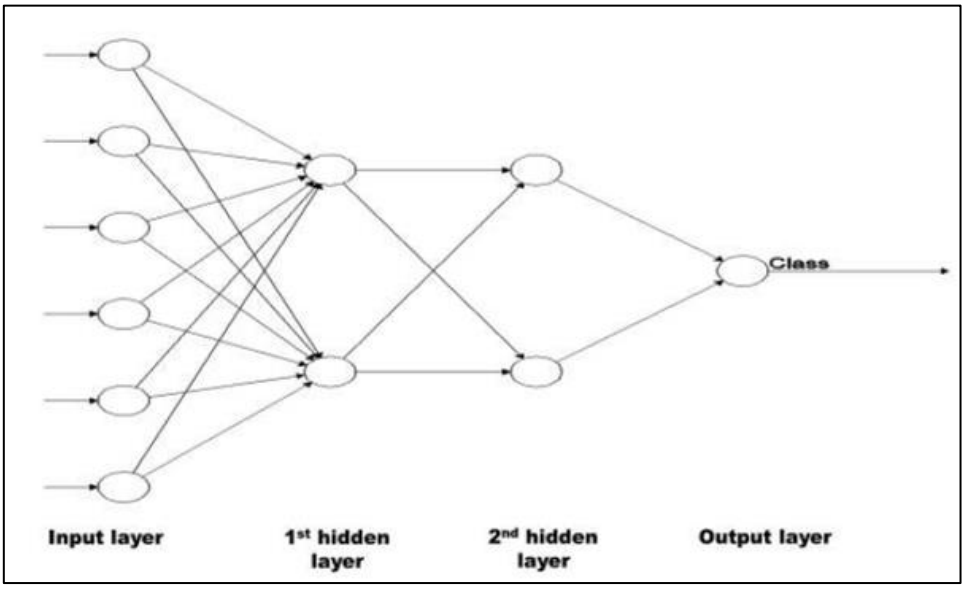


Figure 2-2 - Feed forward neural network with 2 hidden layers

Neural networks with multiple hidden layers will be the deep learning models that will be considered throughout this text. Figure 2-2 shows the simplest form of deep neural network, which contains at least two layers of hidden neurons. Each additional layer in this model produces the output by utilizing the previous layer as input. Deep multi-layer neural networks constitute of many levels of non linearities, allowing them to represent nonlinear functions compactly. They are excellent at identifying complicated patterns from a set of data and have been utilized to improves aspects such as natural language processing and improving computer vision. In addition, it's also been used to solve unstructured challenges that have to do with data.[9]

2.2 Neural Networks

2.2.1 The Neuron

The Human Brain Neuron

Deep learning was created in an attempt to mimic the complex human brain and its ability to learn and observe. And so, deep learning algorithms are structured upon the understanding of the basic building block of the human brain; the neuron. The Neuron is a simple body that, when is connected in a huge neural network, makes this network perform extremely complex tasks. A Neuron's main function is to receive and transmit chemical or electrical signals from and to other neurons connected end-to-end in the same neural network. Neuron's has Dendrites, which are responsible for receiving a signal, and Axons, which are responsible for transmission of a signal (as show in figure 2-3). [9], [10]

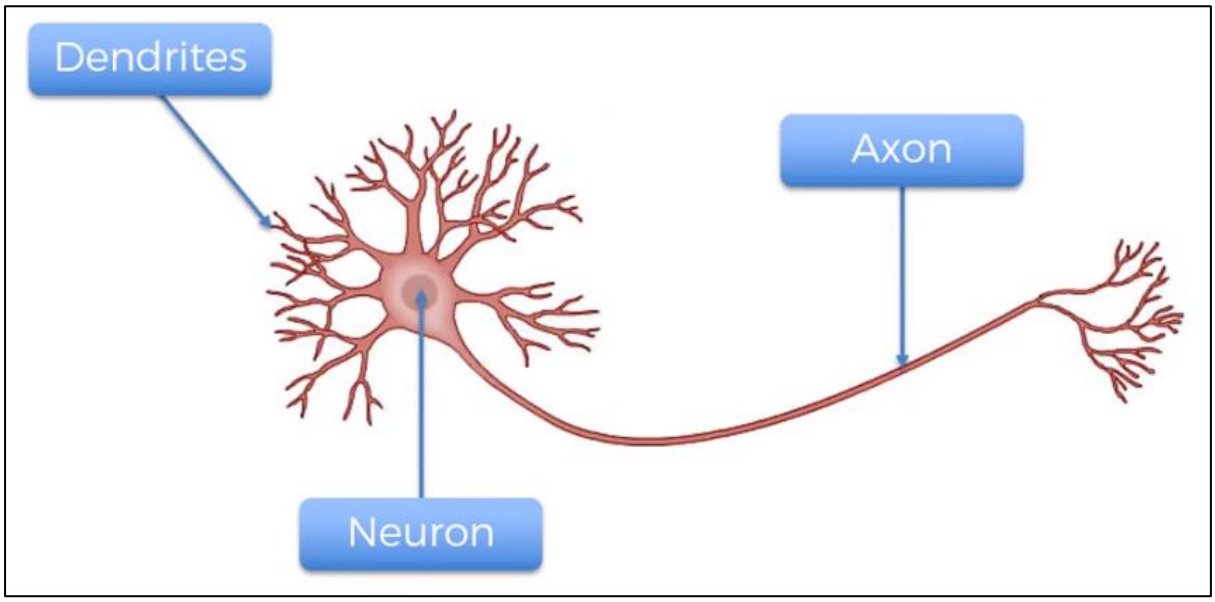


Figure 2-3- The Human Brain Neuron

Signals are passed from one neuron to another via what is called an "action potential". It is a spike in electricity along the cell membrane of a neuron. The interesting thing about action potentials is that either they happen, or they don't. That is why neurons were interesting to computer scientists. This state of being on or off, is just how computers work; 1 or 0. [10]

The Artificial Neuron

As mentioned earlier, the artificial neuron mimics the biological one. Signals come in as input, some processing occur in the neuron itself, then signals come out as output (as shown in figure 2-4). The most simple implementation of a neural network is a Perceptron. A Perceptron follows the *feed-forward* model, where inputs are sent into the neuron are processed and then result in an output. [10]

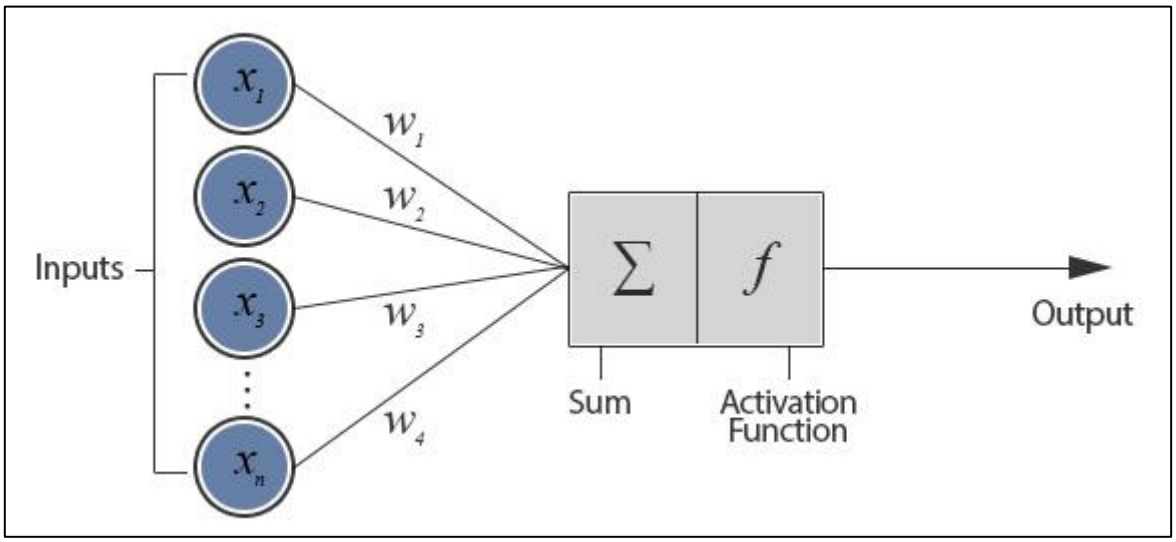


Figure 2-4 - The Simplest Neural Network – The Perceptron.

A Perceptron neural network follows the following steps:

1. Receive inputs.
2. Weight Inputs.
3. Sum Inputs.
4. Generate Output.

A Perceptron will typically begin by assigning random weights. Each input is multiplied by its weight. Those values are then summed and passed through an activation function, which is explained in the next section.
There is one more aspect to the Perceptron and that is ***bias.*** Biases are put into neural networks as one last input to avoid a result summation of zero. [9]

2.2.2 Activation Functions

In neural networks, activation or transfer functions create bounds for the output of neurons. Neural networks can use various activation functions.

Choosing an activation function for your neural network is an essential consideration because it can affect how the input data should be formatted.

Linear Activation Function

The linear activation function is the most simple transfer function that can be used in a neural network. It simply returns the value that the input neuron has passed to it. Regression neural networks will typically use the linear activation function to output a numerical value (figure 2-5). [10]

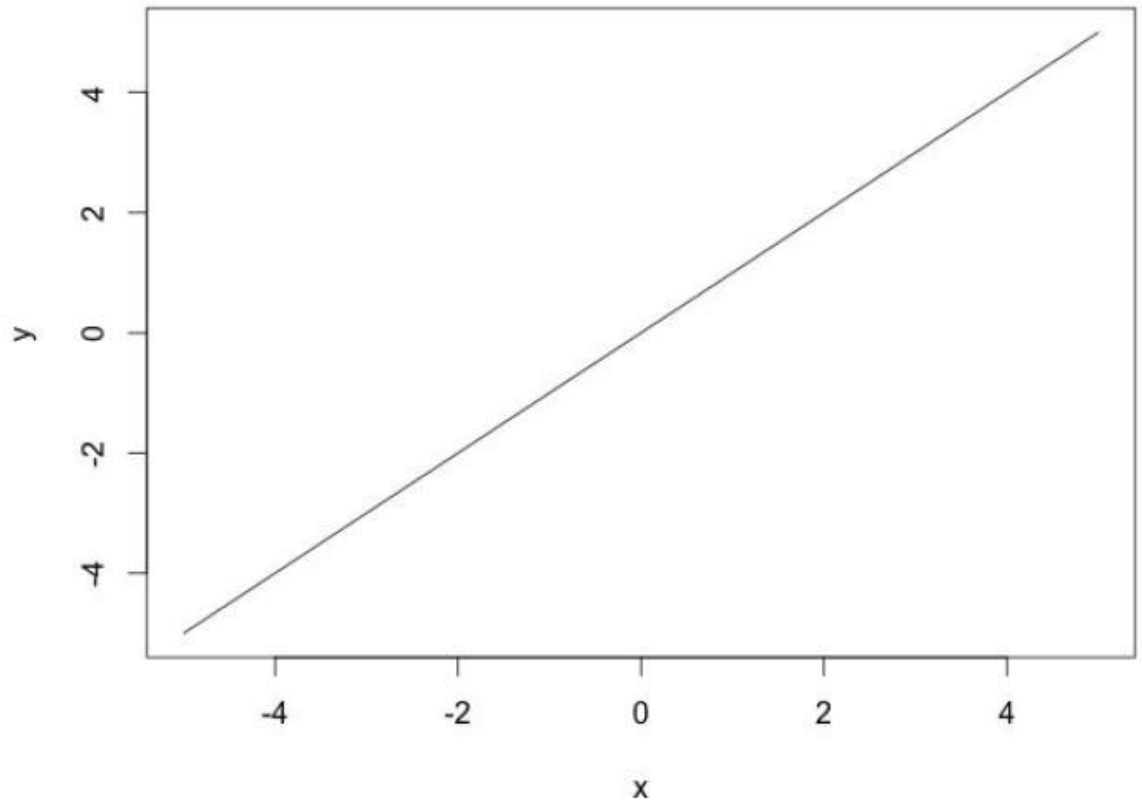


Figure 2-5 Linear Activation Function

Sigmoid Activation Function

The sigmoid or logistic activation function is a typical choice for feed-forward neural networks that only output positive numbers. It is used to ensure that the output values remain within a small pre-defined range (figure 2-7).[10]

$$\phi(x) = \frac{1}{1 + e^{-x}}$$

Figure 2-6 Sigmoid Activation Function Equation

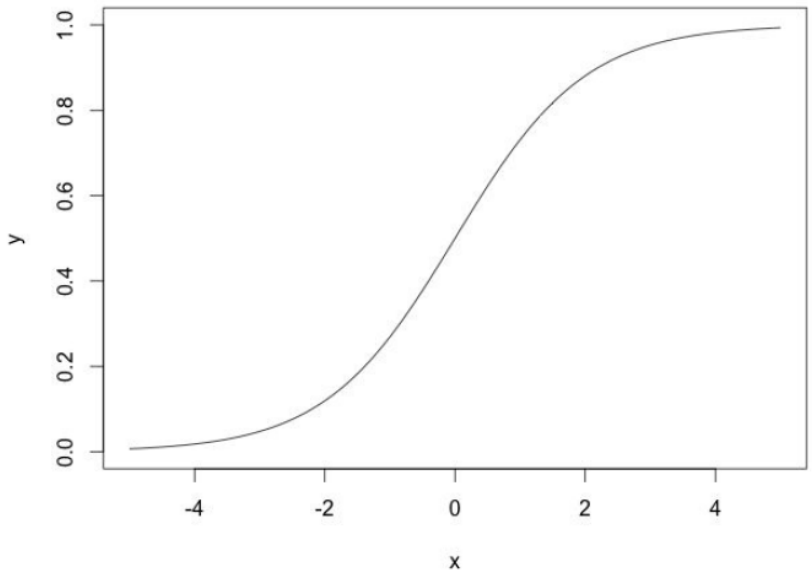


Figure 2-7 Sigmoid Activation Function Graph

SoftMax Activation Function

The SoftMax activation function is typically used on a classification neural network. In such neural networks, there are pre-defined classes that each of the inputs should be categorized to. The neuron with the highest numerical value claims the input within its class. [10]

$$\phi_i = \frac{e^{z_i}}{\sum_{j \in group} e^{z_j}}$$

Figure 2-8 The SoftMax Activation Function Equation

I will use the SoftMax activation function later in chapter 4 to classify hand written digits to their pre-defined classes from 0 to 9.

Rectified Linear Units (ReLU) Activation Function

Most neural networks utilize the ReLU activation function on its hidden layers while either SoftMax or linear functions on the output layer. The ReLU function basically removes the negative part of the function. [10]

The ReLU function has the following equation and graph.

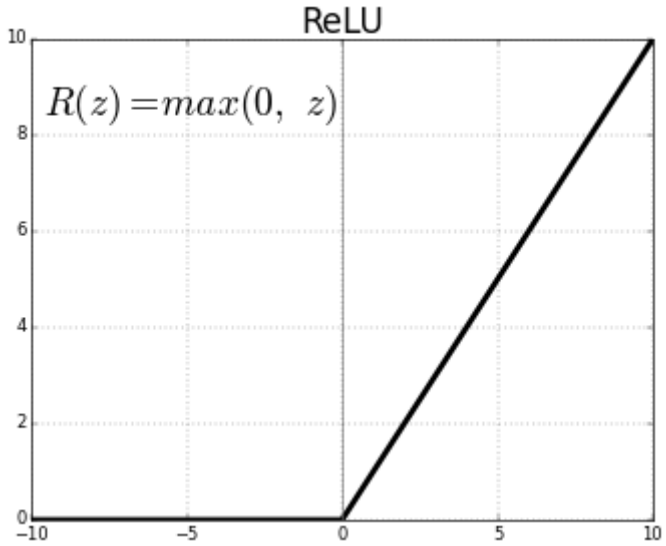


Figure 2-9 - ReLU Activation Function

2.2.3 Backpropagation

Backpropagation is an essential concept in neural networks. It was introduced by Rumelhart, Hinton, & Williams (1986) and is still very widely used today. Backpropagation is a type of gradient decent, which is the computation of a gradient on each weight in a neural network for each training element. As neural networks are not expected to output the expected values from the first feed forward run, the gradient of each weight will give an indication about how much the weights need to be adjusted to achieve the expected output. In a case when the output is exactly what was expected then the gradient of the weight would be 0, meaning that there is no change needed to adjust the weights.

The gradient is the derivative of the error function at the weight's current value. The error function is present in the algorithm to measure how far are the current

output values from the expected ones according to the already labelled data. As the neural networks trains, optimization occurs and error is minimized.
To sum up, when a neural network is initialized, random weights are assigned to inputs then fed to the neurons. Then, those values are summed and enter the activation function then sent to the next layer. The outermost layer's output is compared to the expected output using an error function. This error function backpropagates the adjustments needed to the weights, reaching a final optimized minimum error (figure 2-9). [9]

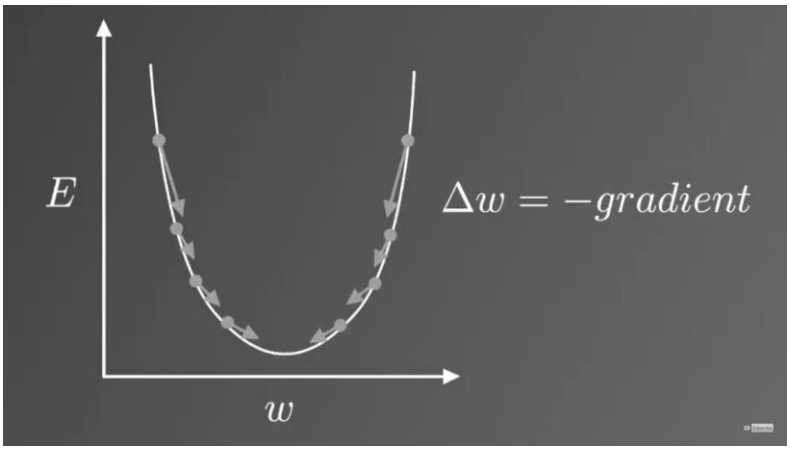


Figure 2-10 - Gradient Decent

2.3 Neural Network Types & Deep Learning Techniques

2.3.1 The Multi-Layer Perceptron

A Multi-Layer Perceptron (MLP) can be regarded as a logistic regression classifier where the input is first transformed by means of a learnt non-linear transformation Φ. This transformation schemes the input data into a space where it becomes linearly separable.[11]

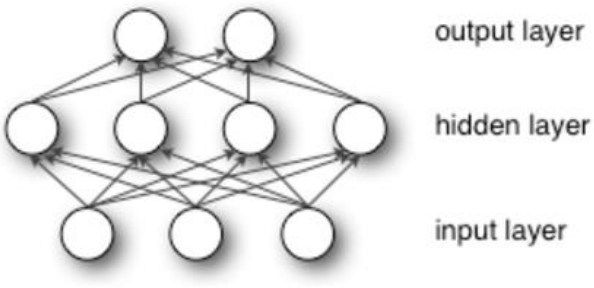


Figure 2-11 - The MLP

The basic scheme of an MLP neural network is shown in figure 4-1. However, the MLP has more than one hidden layer.

2.3.2 Convolutional Neural Networks

Convolutional neural networks or CNNs or ConvNets are a kind of feed-forward ANN in which the connection between its neurons is inspired by the form found in the animal visual cortex. Individual cortical neurons respond to incentives in a region known as the receptive field. Those receptive fields of various neurons overlap such that they tile the visual field. The response of an individual neuron to stimuli within its receptive field can be approximated mathematically by a convolution operation. CNNs are mainly used in computer vison or natural language processing fields.

A ConvNet arranges its neurons in three dimensions (width, height, depth), as visualized in one of the layers. Every layer of a ConvNet transforms the 3D input volume to a 3D output volume of neuron activations. In this example, the red input layer holds the image, so its width and height would be the dimensions of the image, and the depth would be 3 (Red, Green, Blue channels)[12].

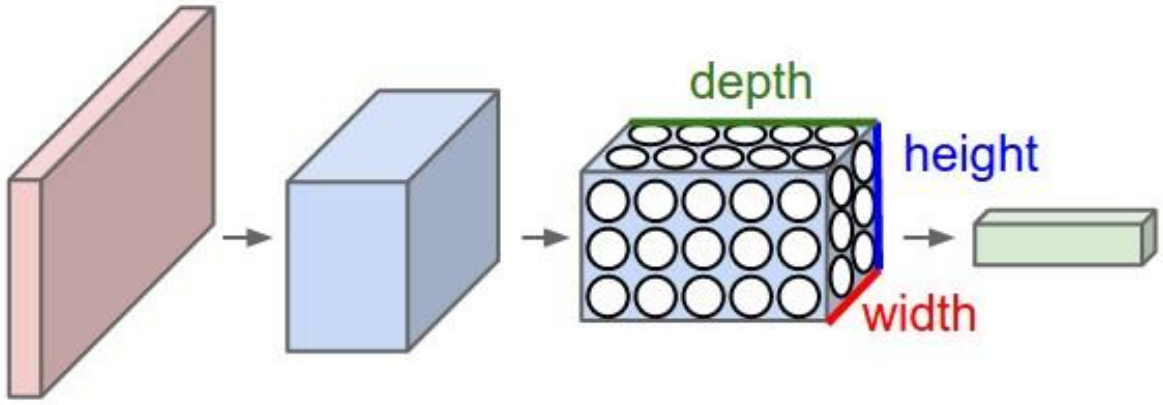


Figure 2-12 CNN for Image Recognition. Adopted from http://deeplearning.ir

2.3.3 Self-Organizing Maps

2.3.4 Restricted Boltzmann Machines

A Restricted Boltzmann Machines (RBM) is an unsupervised learning model that approximates the probability of sample data. The “restricted” part of the name comes from the fact that there are no connections between units in the same layer. The parameters of the model are learned by maximizing the probability function of the samples. Since it is used to approximate a probability density function it is often referred to in the literature as a generative model. Generative learning involves making guesses about the probability distribution of the original input in order to reconstruct it.

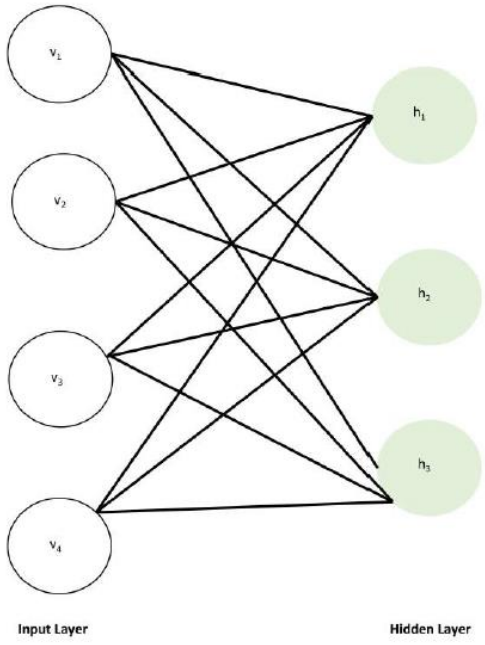


Figure 2-13 – RBM -

Figure 2-13 shoes a graphical presentation of an RBM, where it consists of 2 layers and there are a number of units with inner layer connections. Connections between layers are symmetric and bidirectional, allowing information transfer in both directions. It has one visible layer containing four nodes and one hidden layer containing three nodes. The visible nodes are related to the input attributes so that for each unit in the visible layer, the corresponding attribute value is observable[9].

2.3.5 Autoencoders

The autoencoder is an unsupervised three-layer feature learning feedforward neural network, shown in figure , it is very similar to the multilayer perceptron; it includes an input layer, a hidden layer and an output layer. The number of neurons in the input layer is equal to the number of neurons in the output layer. The number of hidden neurons is less (or more) than the number of input neurons. It is different from an MLP because the output layer has as many nodes as the input layer, and instead of training it to predict some target value y given inputs x, an autoencoder is trained to reconstruct its own inputs x.

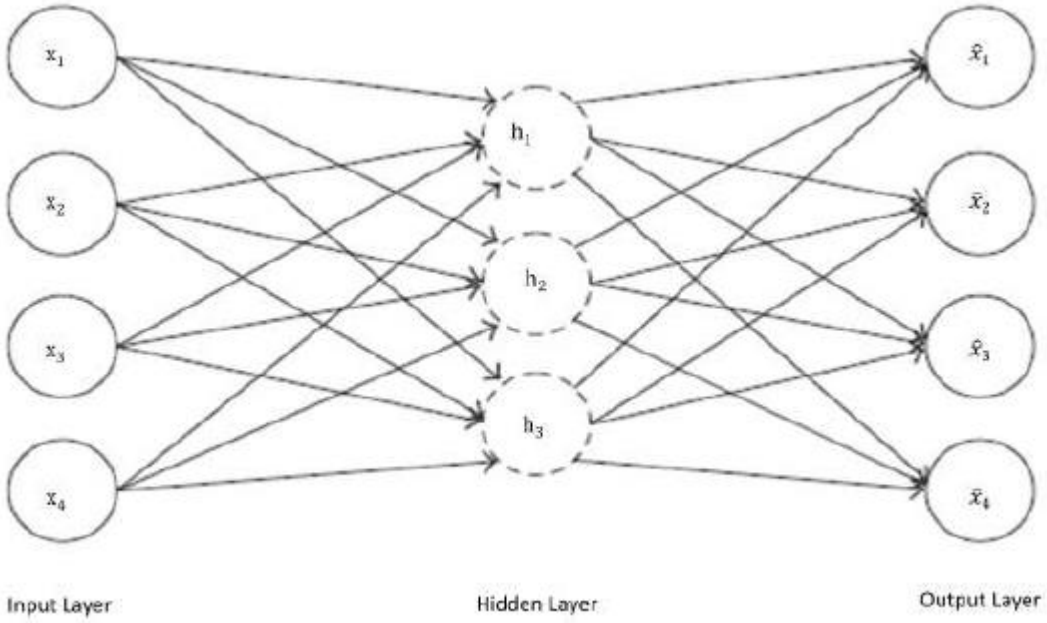


Figure 2-14 - Autoencoder

The autoencoder consists of an encoder and a decoder. The mapping of the input layer to the hidden layer is called encoding and the mapping of the hidden layer to the output layer is called decoding. The encoder takes the vector of input attributes and transforms them typically through sigmoid activation functions in the hidden layers into new features. The decoder then converts these features back to the original input attributes.

2.4 Deep Learning Tools

2.4.1 Libraries & Frameworks

2.4.2 Cloud Computing Solutions

The Cloud Computing Market

Cloud computing has solely changed the way technology works. The ability to host a website on the cloud, manage files and perform actions that were impossible in the past are all results of the presence of cloud computing. Although many new users tend to only look at the cost when it comes to choosing a cloud platform, several business needs need to be taken into account to make a final decision. The three major cloud platforms are compared in this research using several factors such as computing power, storage, databases, location and documentation[13].

The main idea and power of cloud computing comes from the fact that it enables companies and giants such as Spotify and Netflix to not worry about infrastructure and the technicalities that come with maintaining a cloud. The three main cloud platforms in the market are Google Cloud Platform (GCP), Amazon Web Services (AWS), Microsoft Azure. Cheap cloud computing prices are nowadays existent as a result of the fierce competition, which helps big businesses capitalize on growth and small businesses grow easily[13].

Introducing the Three Big Players

The first major player in the market is **Google Cloud Platform (GCP)**. The initial introduction in the market was done by Google to power their own services: Google and YouTube. Enterprise services were later on built enabling anyone to host in the cloud. The second major player is **Amazon Web Services**, referred to as AWS. AWS is one of the longest operating cloud platforms in the market. The have extensive computing services and essential cloud aspects such as mobile networking and deployment are covered. The third and final cloud

platform being compared is **Microsoft Azure**. Being almost as old as GCP in the market, Azure also provides a complete set of cloud services.

Comparison Across Parameters

Compute & Processing Power

The processing power that cloud services offer is referred to as compute. The more compute power, the better. All three major players offer compute power at various levels.

Amazon Web Services (AWS) offer EC2(Elastic Compute Cloud) which is capable of handling all compute services. EC2 works by managing virtual machines that come with preconfigured settings to facilitate use or can be configured by the owner. On the other hand, Google Cloud Platform (GCP) offers Google Compute Engine (GCE) to perform the same essential operation. In addition, Microsoft Azure offers Virtual Machines and Virtual Machine Scale Sets. All three players support containers and are easy to manage and are portable. Users can add more stats or move them to new locations easily.

Although all three players are similar when it comes to compute power, the real difference lies in comparing price and user experience, which is what users would take into account when choosing a cloud platform according to compute power[13].

Storage

Storage is an essential parameter when it comes to understanding and analysing how a cloud platform operates. The way cloud storage operates is entirely different than the normal SSD on our day-to-day computers as multiple issues need to be solved and it also needs to ensure that no data is lost during transfers.

Amazon Web Services (AWS) offers S3 (Simple Storage Service), which is extensive in community support, documentation, etc. and is considered a proper storage solution with plenty of resources. On the other hand, Google

Cloud Platform also offers Google Cloud Storage and Azure offers Microsoft Azure Storage which are considered reliable services. Although all three players offer similar storage solutions, Amazon is considered to have the better storage solution but are more costly[13].

Locations

Location is essential when it comes to deploying the application. Making sure the application performs at its peak by having the lowest possible route to the intended customer base is crucial. All three major players offer great coverage worldwide when it comes to location. However, Amazon leads as it has 42 availability zones. Google Cloud Platform has a presence across 33 countries, and Microsoft Azure covers 32 regions. New regions are added on a regular basis by all three players[13].

Databases

Software terms and conditions and how each cloud platform service manages data on their cloud is quite essential and should be taken into account when investing in any of the platforms.

Given that database images from different vendors enables a user to start more easily, Google seems to be slacking when it comes to providing this option to the end user. On the other hand, Amazon Web Services and Microsoft Azure both offer more several options for the end user to work with. Amazon's RDS (Relational Database Service) provides support for Oracle and other major databases, as their services manages everything from updating to patching to offering solutions to common database issues. Similarly, Cloud SQL offers SQL database handling features for Google Cloud Platform, and Azure SQL offers the same for Microsoft Azure. Both also offer high-performance database choices[13].

Documentation

Documentation is the last parameter used for comparing the three major cloud platform players in this research. Documentation is essential when it comes to choosing the appropriate cloud platform for a user as ease of use is a very valid factor that should be taken into account. Amazon Web Services offers the best documentation, followed by Microsoft Azure and Google Cloud Platform. AWS's strong documentation features comes from it's age in the market and contribution by various people over the course of a decade[13].

Conclusions

From my experience and my research of the three top cloud computational platforms. I chose to use Amazon Web Services due to its generous free credits given to students. I have received 45$ worth of credits that has enabled me to use their GPUs for up to 60 hours. Also, another reason is their marketplace that has 3rd party provided solutions on top of Amazon's cloud virtual computers. Amazon provides Linux virtual machines, but the market place has 3rd party provides that offer Linux virtual machine tailored for machine learning as they come with Python, TensorFlow and NvidiaConda GPU drivers pre-installed so that the user can focus on testing and deployment of their research.

3 Web Scraping

3.1 Introduction

3.1.1 What is Web Scraping and when do we need it?

After we discussed the importance of data in the AI world. It's time to deep dive into one of the most prominent methods of attaining data. The Internet contains massive amounts of data. Most of this data is free for anyone to access. However, most of the time, this data is embedded within the structure of webpage which makes it hard to attain. Also, there is the manual side of attaining this data. Say you want to attain a data set of 1000 images for training your AI model. You can right-click-save-as every and each image, but how much time and effort would that take. For those two reasons, we need Web Scraping to programmatically extract the data we need as we program them. [14]

On the other hand, in a perfect world, people wouldn't actually need to use Web Scarping to extract data from the web as webpages -in a perfect world- provide APIs to share their data with anyone who needs them in an organised manner. In the real world, in fact, many websites do provide APIs to access their data but they also put restrictions of quotas on the frequency of requests as well as the amount of data being accessed. So this means that we cannot always rely on APIs to get our needed data, and that is why we use Web Scraping. [14]

3.1.2 The Legality of Web Scraping

Web Scraping is still a new field that is taking its first steps. And so, laws and rules are still vague and unclear. However, the rule of thumb would be that if the data you are scraping will be used for personal use, then it is okay. On the contrary, if the data is going to be republished, then the scraper needs to be caution with the original publisher's rules. All scrapers, though, need to scrape the web politely. This is done, to name a few, by defining a *user agent* to

identify the person scraping and also by sending download requests at a reasonable rate. [14]

3.2 Web Scraping with Python

Figure 3-1Python Logo

What is Python?

Python is a high-level, general purpose, object-oriented language. High-level means that the programming language is further from the computer's machine language (like Assembly) and closer to the human communication languages[15]. Other examples of high level languages would be C++ and Java. High-level languages, especially Python, are highly readable, shorter and take less time to write. Object-oriented means that the programming language follows the logic of how we think as humans about the program that we are writing [16].

That is why I chose Python for Web Scraping and also for the deep learning later.

3.2.1 BeautifulSoup

BeautifulSoup is a Python library that parses a web page's HTML then lets the programmer easily navigate within. BeautifulSoup is considered the easiest method of Web Scraping. It will lay the basis that will ease the understanding of the main Web Scraping platform that I will use; Scrapy. [14]

I will jump directly into an example then analyse it. The example below will scrap the titles of vacancy postings from Jobs.cz. The webpage looks as follows:

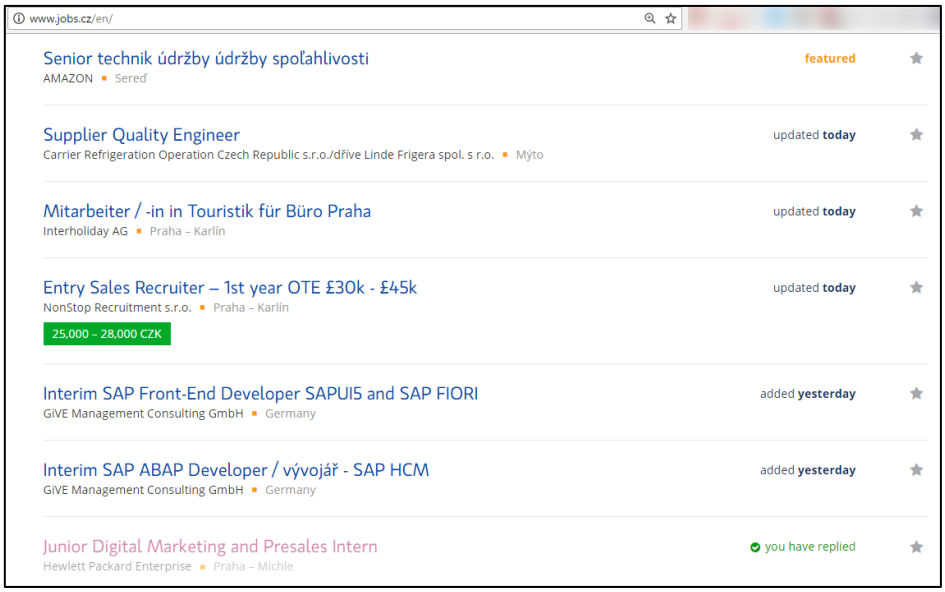


Figure 3-2- Jobs.cz to be scraped

Our code looks as follows:

```
from bs4 import BeautifulSoup
import requests

headers = {'user-agent': 'Mozilla/5.0 (Windows NT 6.1; Win64; x64)'+
    'AppleWebKit/537.36 (KHTML, like Gecko) Chrome/55.0.2883.87 Safari/537.36'}

url = 'http://www.jobs.cz/en/'
r = requests.get(url)
html = r.text
soup = BeautifulSoup(html, "lxml")
for item in soup.select('.search-list__main-info__title'):
    print(item.text + '\n')
```

```
Senior technik údržby údržby spoľahlivosti

Supplier Quality Engineer

Mitarbeiter / -in in Touristik für Büro Praha

Entry Sales Recruiter – 1st year OTE £30k - £45k

Interim SAP Front-End Developer SAPUI5 and SAP FIORI

Interim SAP ABAP Developer / vývojář - SAP HCM
```

Figure 3-3- Web Scraping Example

Let me explain each line of the code that we used:

1. I imported the 2 libraries that we are going to use; *requests* and *BeautifulSoup*. Requests is the library used to make a request to *get* a web page.

2. I assigned declared the variable *header* and I assigned something called user-agent in it.
    - User agents tells the server or the website owner details about you so that you are less likely to be considered a malicious bot and be blocked.
3. I assigned the URL to be scraped in the variable *url*.
4. I used requests library's *get* method to make a request to the URL.
5. I used request library's *text* method to return the HTML document as text and stored it in the variable *html.*
6. I used BeautifulSoup's main method called basically beautifulsoup to store the parsed the HTML document returned by requests in the variable *html* into the new variable *soup*.
7. Now that the variable *soup* has the parsed HTML, we can navigate through the HTML easily using BeautifulSoup's other methods. Here we created a *For* loop iterated by the *find_all* method that finds all matches for *".search-list__main-info__title"* and for each match the for loop prints the outcome of the finding.

So, what is *".search-list__main-info__title"* and how did I know that it contains the titles of the job postings? This is why I introduced CSS and HTML earlier. I will explain in the upcoming section.

Selecting elements in an HTML document

There is a huge amount of ways to select and find elements within an HTML webpage. Depending on the scraping task itself, the selection may be based on a very complex expression or as simple as the one used above. The one that I used above, simply selects any HTML element within the document that has a class attribute equal to *"search-list__main-info__title".* The *"."* used in the beginning in our code is just to say that the following is a class. We can use "#"

if we are selecting based on ID not class. Let's see how to know the class or ID of the item you want to scrap.[14]

Normally, people would view a webpages source and manually find the elements. But that is hectic and inefficient. That is why we have the Google Chrome Developer tools and Firebug, if you are using Firefox. So, I by right clicking the item that I need to scrape and choose *Inspect* (on *Chrome).* I get the below:

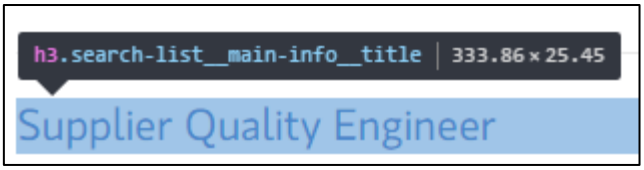


Figure 3-4- Finding an element's class in HTML

This basically means that the title you are interested in is wrapped inside <h3> tags and has a class of *"search-list__main-info__title".*

3.2.2 Scrapy

Scrapy a Python web framework for scraping data from the web. Scrapy was designed to scrape huge amounts of data while handling most of the manual work that libraries like BeautifulSoup does not. It features also a robust system of storing data both locally and remotely in databases as well as other in all of the major data formats available today. For example, if the user wants to scrape product listings from website that contains 16 different web pages, Scrapy will allow them to perform those requests all at once simultaneously. If scarping a page takes a normally takes one second, then you'd need 16 seconds to perform that task. But scrapy handles them all in one second. If each page has 100 listings, then 1600 requests are needed to be made. Moreover, if the user is storing the data into the cloud, which for example would take 3 seconds, that is 4800 write requests all performed in parallel. [17]

Furthermore, scrapy has a huge community with a very clear and conside knowledge base, which is a major advantage when starting to work with a new

framework. Also, Scrapy handles and understands broken HTML. I provided examples how BeautifulSoup selects elements using selectors from web pages. But Scrapy has a higher level interface called XPath which I will explore in the upcoming section.[17]

Scrapy Project

Starting a Project

To start a new Scrapy project, the following is typed in the command line or terminal with *properties* being the project's name:

```
$ scrapy startproject properties
```

Figure 3-5 - Scrapy Project Initiation

The file tree structure goes as follow:

```
properties/
       scrapy.cfg                deploy configuration file
       properties/               project's Python module, you'll import your code from here
              __init__.py
              items.py           project items definition file
              pipelines.py       project pipelines file
              settings.py        project settings file
              spiders/           a directory where the spiders are put
                     __init__.py
```

Figure 3-6 - Scrapy Project File Structure

I will now demonstrate Scrapy with a practical example that I have built to scrape Amazon.com for images to be used for deep learning later.

Scraping Amazon.com for Images

I have used Scrapy to scrape 16000 images off images, I will explain the procedure in this section.

Exploring settings.py

Settings.py gives users a lot of functionality. Using pre-defined functions already embedded with Scrapy, the user can just adjust those settings to tweak

their scrape. The table below shows the most important settings and their description.

ROBOTSTXT_OBEY	If enabled, Scrapy will respect robots.txt policies. It is a way of web scraping politely.
DOWNLOAD_DELAY	Adjust the time between 2 subsequent downloads. Again, to protect the servers.
AUTOTHROTTLE_ENABLED	This option, if enabled, would let Scrapy handle the speed of download traffic according to the Scrapy server and the website being crawled. So, mitigating a lot of problems for the user and the website server.
USER-AGENT	A part of being polite when scraping a webpage. A variable that defines the scraper, their email and the project they're working on.

Table 1- Scrapy's Settings.py

Items.py

This file has a single class that contains any item that is being considered for scraping. It could by an image, a text listing, a video, etc.

Here the field is of the URL of the images of shoes to be downloaded.

```
class AmazonspiderItem(scrapy.Item):
    # define the fields for your item here like:
    image_urls = scrapy.Field()
```

The Spider

Spiders is just another name for web crawlers. And web crawling is the act of swiftly moving through the web in a programmatic manner, while scraping occurs when there is data handling involved.

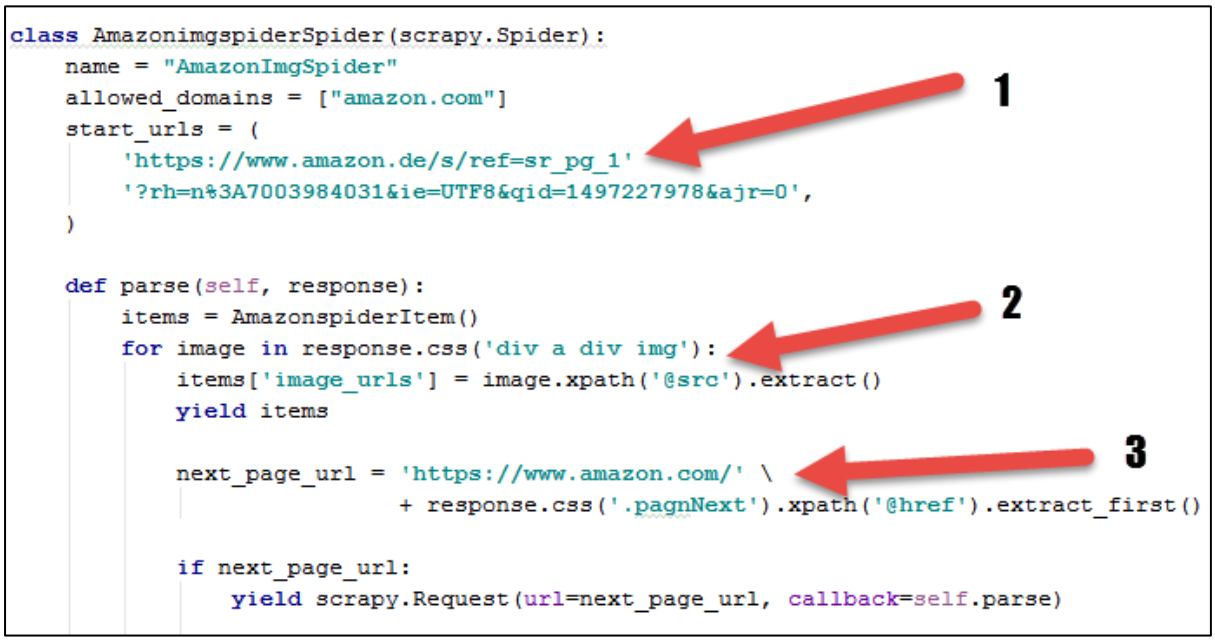


Figure 3-7- Amazon Spider

As simple as it seems, the whole spider that allowed me to download 16000 images consists (as shown in figure 3-7) of a class that has 3 small parts.

`1.start_url:`

This is just the starting URL where the crawl needs to be started. In my case, it is the URL of the first women shoes product listing.

`2.for loop, downloading images:`

As explained before in the BeautifulSoup section, here we use CSS selectors to select the image elements. This for loop basically creates a list of all the <***img***> elements that are inside a <***div***> element that are inside an <***a***> element that are inside a <***div***> element. To put that simply, it is just as written in the code: "**`div > a > div > img`**". The procedure for obtaining that path is shown in the figure below.

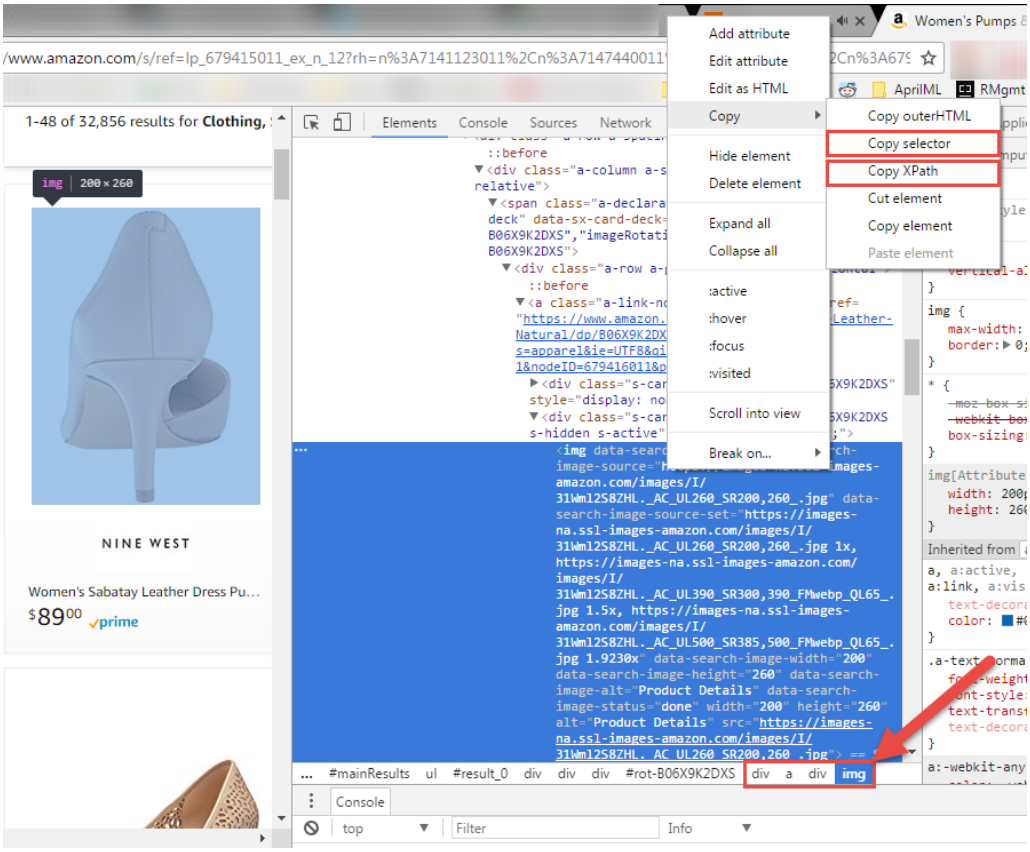


Figure 3-8 - How to obtain a CSS Selector

A list has been created of all the image elements, but the images has not been downloaded to this point. The previously mentioned items.py had a field called ***image_url***. It is time to use this field to cast the image URLs into. Here I used the XPath Selector (the feature that Scrapy is very popular for having) method to obtain the URL.

```
items['image_urls'] = image.xpath('@src').extract()
```

This line means: for each "image" (the for loop index) in the list of <***img***> elements, extract using XPath, the ***@src*** of the currently selected element which returns a URL. This URL for each element is appended into items['**image_urls**'].

```
▼<div class="s-card s-card-group-rot-B06X9K2DXS
s-active" style="display: block;">
    <img src="https://images-na.ssl-images-
    amazon.com/images/I/
    310NyCB6KWL._AC_UL260_SR200,260_.jpg" srcset=
    "https://images-na.ssl-images-amazon.com/
    images/I/310NyCB6KWL._AC_UL260_SR200,
    260_.jpg 1x, https://images-na.ssl-images-
```

Figure 3-9 - img src

3. for loop, next_page_url:

After obtaining all image URL the crawler should then open the next page.

```
▼<span class="pagnRA">
  ▼<a title="Next Page" id="pagnNextLink" class=
  "pagnNext" href="/s/ref=lp_679416011_pg_2?
  rh=n%3A7141123011%2Cn%3A7147440011%2Cn%3A679337011%
  2Cn%3A679416011&page=2&ie=UTF8&qid=1497523101">
      <span id="pagnNextString">Next Page</span>
      <span class="srSprite pagnNextArrow"></span>
    </a>
  </span>
```

Figure 3-10 - grabbing next page's URL

Luckily, Amazon has a specific class and ID attributes for the next page's link. I utilized that below.

```
next_page_url = 'https://www.amazon.com/'
        + response.css('.pagnNext').xpath('@href').extract_first()
```

This line of code utilizes the 2 methods of selection. First the element having the class attribute= **pagnNext** is grabbed using CSS selectors, then within that element the **'@href'** is grabbed using XPath.

3.2.2.1 The Resulting Data Set

The above spider has resulted in scraping 16,000 images off Amazon.com and storing them locally. This data set will be utilized later for deep learning.

Figure 3-11 - Scraped Images

4 Applying Deep Learning for ORI

4.1 MINST – Multi-Layer Perceptron with 2 hidden layers

I will be using in this example the MNIST data set. This data set consists of a training set of 60,000 labelled hand written digits and another 10,000 as test set of hand written digits.

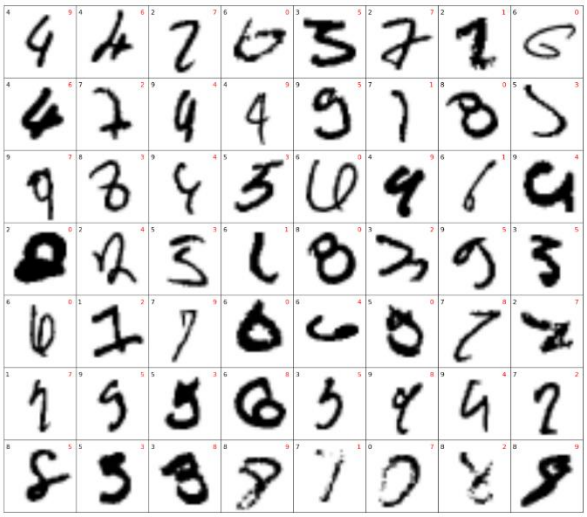


Figure 4-1 - MNIST Data Set

The images are black and white images of size 28 x 28 pixels, or 784 pixels total. Our features will be the pixel values for each pixel. Either the pixel is "white" (blank with a 0), or there is some pixel value.

I will try to correctly predict what number is written down based solely on the image data in the form of an array.

Working with the MNIST data set in Deep Learning is the equivalent of printing “Hello World” when starting to learn a new programming language.

Importing TensorFlow & MNIST

This data set is so popular to the extent that it is embedded within TensorFlow.

```
import tensorflow as tf

# Import MINST data
from tensorflow.examples.tutorials.mnist import input_data
mnist = input_data.read_data_sets("/tmp/data/", one_hot=True)
```

```
Extracting /tmp/data/train-images-idx3-ubyte.gz
Extracting /tmp/data/train-labels-idx1-ubyte.gz
Extracting /tmp/data/t10k-images-idx3-ubyte.gz
Extracting /tmp/data/t10k-labels-idx1-ubyte.gz
```

Figure 4-2 - Importing MNIST

Visualising a Sample

First, I reshaped the vector into a 28*28 matrix. As mentioned before, the data set consists of flattened 28*28 = 784 vectors. The reshape is made to visualise the image in 2D.

The Matplotlib library (imported below) is essential to visualising data in Python, but it is beyond the scope of this paper.

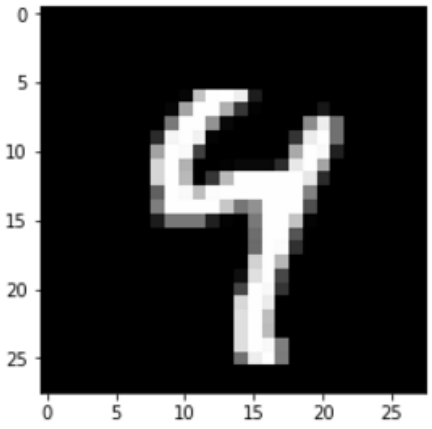
```
sample = mnist.train.images[2].reshape(28,28)
```

```
import matplotlib.pyplot as plt
%matplotlib inline
```

```
plt.imshow(sample,cmap="gray")

#Clearly a Four
```

```
<matplotlib.image.AxesImage at 0x7ff4b6c19cf8>
```



Determining the deep learning parameters

As mentioned, this data set is very popular and so, the optimum parameters for initialising a deep learning using this data set are already known. However, I will tweak those parameters after the first trial and analyse how it affects the accuracy of the model's predictions.

These parameters are:

- Learning Rate - How quickly to adjust the cost function.
- Training Epochs - How many training cycles to go through
- Batch Size - Size of the 'batches' of training data

```
learning_rate = 0.001
training_epochs = 15
batch_size = 100
```

Figure 4-3 - Learning Parameters

Network Parameters

These parameters define our neural network. They vary according to how the data looks like.

4. *n_hidden_1*: Number of features in the first layer
5. *n_hidden_2*: Number of features in the second layer
6. *n_input*: Data shape (i.e. L * W pixels)
7. *n_classes*: Data total number of classes
8. *n_samples*: Training samples

```
n_hidden_1 = 256 # 1st layer number of features
n_hidden_2 = 256 # 2nd layer number of features
n_input = 784 # MNIST data input (img shape: 28*28)
n_classes = 10 # MNIST total classes (0-9 digits)
n_samples = mnist.train.num_examples
```

Figure 4-4 - Network Parameters

TensorFlow Graph Input

```
x = tf.placeholder("float", [None, n_input])
y = tf.placeholder("float", [None, n_classes])
```

- Any TensorFlow computation is represented as a dataflow graph.
- A placeholder is simply a variable that we will assign data to later.

Multi-Layer Model

```
def multilayer_perceptron(x, weights, biases):
    '''
    x : Place Holder for Data Input
    weights: Dictionary of weights
    biases: Dicitionary of biases
    '''

    # First Hidden layer with ReLU activation
    layer_1 = tf.add(tf.matmul(x, weights['h1']), biases['b1'])
    layer_1 = tf.nn.relu(layer_1)

    # Second Hidden layer with ReLU activation
    layer_2 = tf.add(tf.matmul(layer_1, weights['h2']), biases['b2'])
    layer_2 = tf.nn.relu(layer_2)

    # Last Output layer with linear activation
    out_layer = tf.matmul(layer_2, weights['out']) + biases['out']
    return out_layer
```

Figure 4-5 - Defining the MLP

Firstly, the input data array is received then is sent to the first hidden layer. Then weights (randomly only initially) will be assigned to the data a and then sent to a node to undergo an activation function along with the Bias. Then it will continue on to the next hidden layer, and so on until the final output layer. In this example, there are just 2 hidden layers. The more the hidden layers the higher the possibility for a more accurate model. However, it is a trade-off between run time and accuracy.

Once the transformed data reaches the output later it is evaluated using the error function that determines how far off is the output from the expected results. In this case, how many classes are correct.

Then, an optimization function is applied to minimize that error and adjust the weights accordingly.

Application of this optimization can be adjusted by altering the learning rate parameter. The lower the rate the higher the possibility of higher accuracy, but it is again a trade-off between accuracy and run time. At some point, lowering the learning rate will not increase accuracy and will reach a reach terminal value.

Weights and Bias

In order for our tensorflow model to work, two dictionaries containing our weight and bias objects for the model need to be created. The tf.variable object is used in this example. This is different from a constant because TensorFlow's Graph Object becomes aware of the states of all the variables. A Variable is a modifiable tensor that lives in TensorFlow's graph of interacting operations. It can be used and even modified by the computation.

```
weights = {
    'h1': tf.Variable(tf.random_normal([n_input, n_hidden_1])),
    'h2': tf.Variable(tf.random_normal([n_hidden_1, n_hidden_2])),
    'out': tf.Variable(tf.random_normal([n_hidden_2, n_classes]))
}
```

Figure 4-6 - Weights Dictionary

```
biases = {
    'b1': tf.Variable(tf.random_normal([n_hidden_1])),
    'b2': tf.Variable(tf.random_normal([n_hidden_2])),
    'out': tf.Variable(tf.random_normal([n_classes]))
}
```

Figure 4-7 - Biases Dictionary

```
# Construct model
pred = multilayer_perceptron(x, weights, biases)
```

Figure 4-8 - Model Construction

Cost and Optimization Functions

Here TensorFlow's built-in functions will be used.

```
# Define loss and optimizer
cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(pred, y))
optimizer = tf.train.AdamOptimizer(learning_rate=learning_rate).minimize(cost)
```

Figure 4-9 - Cost and Optimization Functions

Initialization of Variables

```
# Initializing the variables
init = tf.initialize_all_variables()
```

Figure 4-10 - Variable Initialization

Running the Session

The session has 2 loops. The outer loop which runs the epochs, and the inner loop which runs the batches for each epoch of training

```
# Launch the session
sess = tf.InteractiveSession()

# Intialize all the variables
sess.run(init)

# Training Epochs
# Essentially the max amount of loops possible before we stop
# May stop earlier if cost/loss limit was set
for epoch in range(training_epochs):

    # Start with cost = 0.0
    avg_cost = 0.0

    # Convert total number of batches to integer
    total_batch = int(n_samples/batch_size)
```

```
    # Loop over all batches
    for i in range(total_batch):

        # Grab the next batch of training data and labels
        batch_x, batch_y = mnist.train.next_batch(batch_size)

        # Feed dictionary for optimization and loss value
        # Returns a tuple, but we only need 'c' the cost
        # So we set an underscore as a "throwaway"
        _, c = sess.run([optimizer, cost], feed_dict={x: batch_x, y: batch_y})

        # Compute average loss
        avg_cost += c / total_batch

    print("Epoch: {} cost={:.4f}".format(epoch+1,avg_cost))

print("Model has completed {} Epochs of Training".format(training_epochs))
```

```
Epoch: 1 cost=156.1939
Epoch: 2 cost=38.7113
Epoch: 3 cost=24.6571
Epoch: 4 cost=17.1834
Epoch: 5 cost=12.6043
Epoch: 6 cost=9.4217
Epoch: 7 cost=7.1025
Epoch: 8 cost=5.3346
Epoch: 9 cost=3.9459
Epoch: 10 cost=3.0107
Epoch: 11 cost=2.2067
Epoch: 12 cost=1.6921
Epoch: 13 cost=1.3159
Epoch: 14 cost=0.9436
Epoch: 15 cost=0.7575
Model has completed 15 Epochs of Training
```

Figure 4-11 – Session Output

Model Evaluations

Tensorflow comes with some built-in functions to ease model evaluate, including ***tf.equal*** and ***tf.cast*** with ***tf.reduce_mean***.

```
# Test model
correct_predictions = tf.equal(tf.argmax(pred, 1), tf.argmax(y, 1))
```

Figure 4-12 - Model Evaluation

In order to get a numerical value for the predictions, ***tf.cast*** will be used to cast the Tensor of Booleans back into a Tensor of Floating point values in order to take the mean of it.

```
correct_predictions = tf.cast(correct_predictions, "float")
```

Figure 4-13 - Getting Numerical Values

Now the ***tf.reduce_mean*** function will be used in order to grab the mean of the elements across the tensor.

```
accuracy = tf.reduce_mean(correct_predictions)
```

Now all what is left is to pass in the actual test data (labelled images) to evaluate accuracy.

The ***eval()*** method lets the user directly evaluate this tensor in a Session without needing to call ***tf.sess()***.

```
print("Accuracy:", accuracy.eval({x: mnist.test.images, y: mnist.test.labels}))
Accuracy: 0.9436
```

Figure 4-14 - MLP's Accuracy

An accuracy of 94% has been achieved. This accuracy can still be increased if the learned rate got decreased and more hidden layers were added to the model.

4.2 Women Shoes Dataset

In this section, I will attempt to build a neural network which aims to cluster images from the dataset of women shoes that I have collected off Amazon.com (chapter 3). This dataset has no labels, meaning that nothing is known about the images. The aim of the neural network is to figure out some kind of pattern that defines the images and cluster similar ones together. This is a problem to be tackled by unsupervised learning techniques.

First, however, I need to define what is similar to me? Is it colour, shape or size? This is a crucial, because the model is based on that decision. Well, shoes comes in all shapes in sizes, and these two criteria do not make them fall in different categories. This means that, two shoes falling in the same class can have 2 different mixture of colours and come in different two different sizes. This leaves the final category; shape. Shape is indeed the only factor that should be of interest.

Figure 4-15 - Shoes Class 1

Figure 4-16 - Shoe Class 2

As we it can be seen from figures 4-6 and 4-7, the images are coloured. What does colour mean in data? Images are just arrays of numbers R * G * B. This means that coloured images are 3 dimensional arrays. But since colour is not a factor in the deep learning model, colour should be eliminated from the images to simplify the problem; dimensionality reduction. Doing this will reduce the dimensions of each array within our data from 3 to 1. The model will be dealing

with greyscale images (below), which does not matter in the learning process, as the only crucial criterion is shape.

Figure 4-17 - Shoes Class 1 – Greyscale

Figure 4-18 - Shoe Class 2 – Greyscale

Limitations to My Own Web Scrapped Dataset

Unfortunately, after extensive research, I have reached the conclusion that working with this dataset would be nearly impossible for reasons that will be mentioned in the last chapter of this paper.

I had the goal to use deep convolutional neural networks to cluster those images utilizing an unsupervised learning technique. The model would find similar images and cluster them together. Then finally I would use Python's data visualization libraries to produce a t-SNE graph where one can easily visualize the clustered classes of shoes and how they relate.

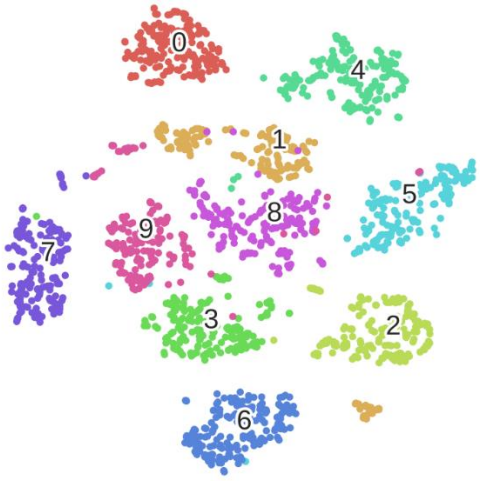


Figure 4-19 - t-SNE Visualization

Unfortunately, as my knowledge of the field increased, I realized that with my current humble knowledge of statistics and mathematics, as well as the lack of available practical tutorials for unsupervised learning. So, I reverted to supervised learning techniques, where I faced another problem. Supervised learning techniques need around an 8:2 training set to test set ratio. This means that I would **need to label 80% of my data manually.** For example, the MINST data set of hand written digits is composed of 60,000 labelled hand written digits and another 10,000 labelled set of hand written digits.

5 Conclusion

After the long journey that I have gladly had during self-studying and researching deep learning, I have realised that the field has gone a long way and has an extremely promising potential to disrupt every and each industry that we have today. However, there is a lot of problems and limitations in the field that I shall discuss below.

Supervised Vs. Unsupervised Learning

I have realised that the supervised learning algorithms and techniques are far more advanced and mature than the unsupervised learning ones. The reason for that is that training neural networks is much easier when we already have labelled data to train the model on and then to assess and optimize it with. On the contrary, unsupervised learning techniques are available and working but they are much more complex and need a very deep understanding of advanced statistics as well as mathematics.

I should mention also that, when it comes to supervised learning, working with one's own datasets is a very hectic process. The user needs to manually label data which contradicts with the goal of artificial intelligence, which is an attempt to let computers learn on their own to ultimately automate tasks with minimal human intervention.

Moreover, people still starting with deep learning will find way more resources and practical tutorials on supervised learning vs. unsupervised.

Noise

Another very crucial observation that I have had about object recognition in images, is the great effect that noise can have on a neural network's predictions.

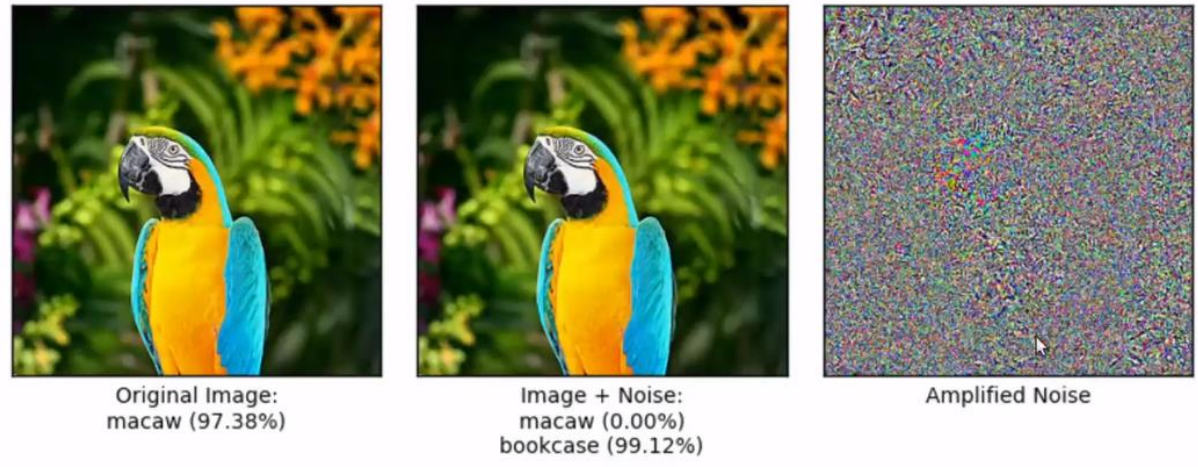


Figure 5-1 - Effect of Noise on ORI

In figure 5-1, noise has been added of only a 3.0 limit per pixel. This is just a mere change of 1.2% per pixel value. In this example, the noise has been added to the Inception Model – A model that has been developed and pre-trained by TensorFlow to recognize and predict the content of images. The user can use the model for predicting images or use it for transfer learning on their own project. Noise has been added in a wat that will alter the prediction of the image to a bookcase. After adding the noise, which is invisible to the human eye, the true prediction (Macaw) decreased from 97% to zero.

I believe that this is a very crucial observation as humans are starting to depend on computer vision for use in self-driving cars, for example. This case shows that an invisible noise to the human eye can alter the vision of the machine entirely, which could lead to drastic life threatening consequences in the future, if not addressed properly.

Appendix

Understanding the Web

As you dig deeper in the world of Web Scraping, you will be impressed by those web browsers that we take for granted and the code that lies underneath those webpages. In fact, the websites that you open on a daily basis and absolutely love, would look hostile and unpleasant without: 1- **JavaScript** which, makes a webpage interactive. 2- **CSS**, that is responsible for formatting the webpage and make it looking nice and welcoming. 3- **HTML**, that describes and defines the content of the webpage.[18]

HTML & CSS

In order to fully understand the fundamentals of Web Scraping, one will to understand the mechanics of code behind a webpage as that is how we are going to access and select exactly the data that we need.

HTML

Some people think that Hypertext Mark-up Language (HTML) is a programming language. In fact, HTML is a mark-up language that outlines the content of the webpage by a series of elements. Each of those elements dictate how the content wrapped inside it is to be rendered in the browser on the user's screen. All the HTML code should be surrounded between <html> and </html> tags. Also, tags like <head> & <body> are essential parts of a webpage's HTML. Let's explore one of the abovementioned elements: the paragraph element, which is defined by the tag <p>. This element defines the content wrapped inside as a **paragraph**.
Now that we understand what an HTML tag is, let's look at a simple HTML code to analyse it.[19]

```
<html>
  <head>
    <title>My test page</title>
  </head>
  <body>
    <h1>
        An Example Page
    </h1>
    <div class="example">
        Some example content is here
    </div>
  </body>
</html>
```

As mentioned before, everything should be enclosed between a <html> and a </html>. The <head> tag contains content that describes things that are not rendered on the screen. It has keywords and information like a page's description that will be useful for search engines to find the webpage.

The <body> element wraps all the viewable content of the webpage inside it. It can contain tags such as <p> (paragraph) <img> (images) (lists) and <a> (links). The <body> element above has an <h1> tag and a tag. The tag simple defines a division or a section within the document while the <h1> tag is simple a heading. Headings in HTML is marked up from <h1> to <h6>[18].

There is something very important to notice, though. There is an attribute given to the abovementioned element called *class*. There are 2 types of attributes that can be given to any kind of HTML element: class or ID. These 2 methods are used to select elements when writing your CSS (To apply style to this particular element) or JavaScript (to trigger that element programmatically)[20].

CSS

As much as HTML is not a programming language, Cascading Style Sheets (CSS) also isn't. However, it is not a mark-up language either. From its name, we can say CSS is a styling language that lets you style the elements included in your HTML documents[21]. Put very simply, if you want to apply some styling to all the paragraph elements in your HTML, you put the following code in your CSS file:

```
p {
    color: blue;
    font-size: 5px;
    text-align: center;
}
```

We can see how CSS defines things like colour, position and background of different elements on a webpage.

For knowing more about HTML & CSS, I suggest opening the tutorials of W3Schools as well as right clicking anywhere in a webpage and navigate *page source*.

Python Crash Course

```
In [3]: age = 24
        name = "Ahmed"

        print('Hellow World! My name  is {}, and I am {} years old'.format(name,age))

        Hellow World! My name  is Ahmed, and I am 24 years old
```

Figure 5-2 - Hello World!

From the example above, we can see how Python is highly readable and very logical to write.

Lists

```
In [45]: [1,2,3] #Lists use [Square Brackets]
Out[45]: [1, 2, 3]

In [48]: ['hey',1,[1,2]] #list within a list
Out[48]: ['hey', 1, [1, 2]]

In [57]: list = ['a','b','c']

In [58]: list
Out[58]: ['a', 'b', 'c']

In [59]: list.append('d') #Append Function: adds an element to the end of the list

In [80]: list
Out[80]: ['a', 'b', 'c', 'd']
```

Figure 5-3 - Python Lists

The Python list object is the most universal classification provided by the language. Lists are ordered collections of randomly typed objects, and they have no fixed size. They are also mutable—unlike strings, lists can be altered.

Indexing and Selection

```
In [62]: list[0] #Index=0, first elements is always 0 not 1.
Out[62]: 'a'

In [65]: list[3] #Index=3, 4th element
Out[65]: 'd'

In [71]: list[1:3] #Starting index=1 (Always indluded) up till but not including index=3
Out[71]: ['b', 'c']

In [76]: list[1:] #Starting index=1 up till the end of the list
Out[76]: ['b', 'c', 'd']

In [83]: list[:2] #Every element in the list uptill but not indluding 2
Out[83]: ['a', 'b']
```

Figure 5-4 - Indexing and Selection in Python

Dictionaries

A dictionary in Python is a list but with a custom index for each element called "key". The key can have any value and it is how you call or select the element.[22]

```
In [4]: d = {'AL':'Alabama','AK':'Alaska','AZ':'Arizona','CA':'California','CO':'Colorado' }

In [5]: d
Out[5]: {'AK': 'Alaska',
         'AL': 'Alabama',
         'AZ': 'Arizona',
         'CA': 'California',
         'CO': 'Colorado'}

In [6]: d['CA']
Out[6]: 'California'
```

If, elseif and else statements

```
In [139]: if 3 == 5: #False
              print('first')
          elif 3 == 3: #True
              print('middle')
          else:
              print('Last')

          middle
```

Loops

```
In [8]: list = [1,2,3,4,5]

In [9]: for item in list: #As easy as you read it. For each item in the list -> Print the item itself- the counter itself.
            print(item)

        1
        2
        3
        4
        5

In [13]: for item in list: #For each item in the list -> Print 'Hi number followed by the counter'
             print('Hello number {}'.format(item))

         Hello number 1
         Hello number 2
         Hello number 3
         Hello number 4
         Hello number 5
```

Functions

```
In [14]: def square(x): #Define a function which outputs the input "x" raised to the power of 2
             return x**2

In [16]: print(square(9))

         81
```

References

[1] B. Marr, 'Why Artificial Intelligence Will Change Our World And Why It Needs To Be Purposeful', *Forbes*. [Online]. Available: https://www.forbes.com/sites/bernardmarr/2016/05/25/why-artificial-intelligence-will-change-our-world-and-why-it-needs-to-be-purposeful/. [Accessed: 28-May-2017].

[2] 'Industrial Revolution - Facts & Summary', *HISTORY.com*. [Online]. Available: http://www.history.com/topics/industrial-revolution. [Accessed: 28-May-2017].

[3] '1926: Field Effect Semiconductor Device Concepts Patented | The Silicon Engine | Computer History Museum'. [Online]. Available: http://www.computerhistory.org/siliconengine/field-effect-semiconductor-device-concepts-patented/. [Accessed: 28-May-2017].

[4] 'Vint Cerf and Bob Kahn, co-inventors of TCP/IP protocol | FierceTelecom'. [Online]. Available: http://www.fiercetelecom.com/special-report/vint-cerf-and-bob-kahn-co-inventors-tcp-ip-protocol. [Accessed: 28-May-2017].

[5] 'John McCarthy: Computer scientist known as the father of AI | The Independent'. [Online]. Available: http://www.independent.co.uk/news/obituaries/john-mccarthy-computer-scientist-known-as-the-father-of-ai-6255307.html. [Accessed: 28-May-2017].

[6] 'A new dawn', *artificial-intelligence*. [Online]. Available: https://www.ubs.com/microsites/artificial-intelligence/en/new-dawn.html. [Accessed: 13-Jun-2017].

[7] Chris Smith, 'The History of Artificial Intelligence'.

[8] 'Data, data everywhere', *The Economist*, 25-Feb-2010.

[9] N. D. Lewis, *Deep learning made easy with R: a gentle introduction for data science*. Place of publication not identified: AusCov, 2016.

[10] J. Heaton and J. Heaton, *Deep learning and neural networks*. St. Louis, MO: Heaton Research, Inc, 2013.

[11] 'Multilayer Perceptron — DeepLearning 0.1 documentation'. [Online]. Available: http://deeplearning.net/tutorial/mlp.html. [Accessed: 15-Jun-2017].

[12] 'CS231n Convolutional Neural Networks for Visual Recognition'. [Online]. Available: http://cs231n.github.io/convolutional-networks/. [Accessed: 18-Jun-2017].

[13] P. on F. 20, 2017 in Connected Devices, Industrial, S. Cities, and Transport, 'Who wins the three-way cloud battle? Google vs. Azure vs. AWS', *ReadWrite*, 20-Feb-2017. [Online]. Available: https://readwrite.com/2017/02/20/wins-three-way-cloud-battle-google-vs-azure-vs-aws-dl1/. [Accessed: 18-Jun-2017].

[14] R. Lawson, *Web Scraping with Python: scrape data from any website with the power of Python*. 2015.

[15] 'What is High-Level Language? Webopedia Definition'. [Online]. Available: http://www.webopedia.com/TERM/H/high_level_language.html. [Accessed: 04-Jun-2017].

[16] 'Introduction to object-oriented languages', *Lynda.com - from LinkedIn*. [Online]. Available: https://www.lynda.com/Programming-Foundations-tutorials/Introduction-object-oriented-languages/83603/90484-4.html. [Accessed: 04-Jun-2017].

[17] D. Kouzis-Loukas, *Learning Scrapy: learn the art of efficient web scraping and crawling with Python*. 2016.

[18] R. Mitchell, *Web scraping with Python: collecting data from the modern web*. 2015.

[19] 'HTML basics', *Mozilla Developer Network*. [Online]. Available: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/HTML_basics. [Accessed: 03-Jun-2017].

[20] 'class', *Mozilla Developer Network*. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTML/Global_attributes/class. [Accessed: 03-Jun-2017].

[21] 'CSS basics', *Mozilla Developer Network*. [Online]. Available: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/CSS_basics. [Accessed: 03-Jun-2017].

[22] 'Organizing data with dictionaries', *Lynda.com - from LinkedIn*. [Online]. Available: https://www.lynda.com/Python-tutorials/Organizing-data-dictionaries/62226/71001-4.html. [Accessed: 04-Jun-2017].