Haaga-Helia
University of Applied Sciences

Artificial Intelligence and auto-generation of code

John Cheng

Bachelor's Thesis
Degree Program in Business Information Technology
13 May 2018

Haaga-Helia
University of Applied Sciences

Abstract

13 May 2018

Author(s)
John Cheng

Degree programme
Business Information Technology

Report/thesis title
Artificial Intelligence and auto-generation of code

Number of pages and appendix pages
6 + 2

2018. At the time this thesis is written, software and especially mobile apps are starting to occupy a huge part of our everyday life. The development team is one of the key-factor deciding the success or failure of a project. Indeed, they are responsible of implementing all the ideas presented by a product owner which are made true by designers.

An app is composed of two parts:

- the Front-End that corresponds to the User Interface, the link between a user and an app
- and the Back-End which is basically the brain, the logic behind every action, every process of an app.

Those working on both are called Full-Stack developers.

Me being part of that category and based on experience, a recurrent problem resides in the time spent to implement the “face” of the app and improve the user experience sometimes at the expense of the “brain”. The result being beautiful apps design-wise but lacking in functionality or practicability.

In order to resolve that handicap, delegating a part of the work to the machine came to mind. Also, Artificial Intelligence is the state of the art technology right now. The goal is to research the possibilities to exploit the processing power and capacity to follow orders of the machine to help developers.

The idea is to feed the machine with a design mockup and based on that, it will be able to recognize and return a code based on the different components and layout found.

By reading this thesis, neophytes will be able to understand the different steps needed to explore a solution with deep explanation of what machine learning is and how to use it for visual recognition.

Furthermore, as machines are extremely good at reproducing, a way to facilitate teams transitions in projects is also detailed.

Finally, a solution is provided where a design will be fed to the Artificial Intelligence and an HTML code is received.

Being a subject relatively new for the time of the thesis, the material and help to implement the final solution are limited. That is why, even incomplete, it should be considered as a boilerplate to continue working on to further improve it.

Keywords
Artificial Intelligence, machine learning, python, code generation, visual recognition

Table of contents

1 Introduction

The Information Technology field is ever-changing. As a matter of fact, the tools used by developers are consistently getting more powerful, helpful, intelligent and thanks to that, the outcome's quality is also increasing.

So, at this stage, what are the problems we have to face? How can we fight against those?

As a Full-Stack developer, I have stumbled upon the hardship of having to implement complex designs and I had to spend hours to make my project look like how it needed to be. Those same hours I could have spent on the back-end part of the project, improving the efficiency of the server-side logic.

And that is when this question came to my mind: "Why am I the one implementing the front-end?"

Through this report, I aim to create a solution in order to bring a new tool to developers. I decided to research how code could be auto-generated by an Artificial Intelligence. The objective would be for the Artificial Intelligence to recognize the different patterns in a given design and to provide the code from it. That said, the way to the solution will go through different stages. First of all, understanding the theory behind the practice is important, so a theoretical part will explain the relevant elements of machine learning as well as the current situation regarding the project, the advantages from such a solution and the philosophy behind it. After, the implementation can start and the three main components of the AI are going to be detailed. And lastly, a conclusion and summary to the thesis will be provided.

The challenges that come with this project are considerable; especially on the implementation side. But I love the problem and I hope you, readers, will love it too.

2 Theoretical framework

The aim of the project is to have an Artificial Intelligence that will be able to detect the different components on a design mock-up and then, based on the detected components, generate a normalized code.

From the project baseline, the following steps are deducted in order to be able to create a working solution: image detection to understand a mockup and thus, determine the components and related properties, generate the code and lastly, blend the two precedent steps together.

Prior to going deeper into the technical explanations, it is important to understand what an Artificial Intelligence is and the current situation regarding this research. After that we will go through how to generate code with an Artificial Intelligence. Then, how the different components of a design mockup are discerned will be discussed.

2.1 Artificial Intelligence (AI)

According to the Oxford dictionary, an Artificial Intelligence is “the theory and development of computer systems able to perform tasks normally requiring human intelligence, such as visual perception, speech recognition, decision-making, and translation between languages”. (Oxford Dictionnary, 2018)

In other words, an AI is a program that can do a task originally made to be done by humans.

The Artificial Intelligence arrived on the big stage in 1956 after a conference at Dartmouth College where the field of AI research was founded. The researches have been through good and bad times when funds were cut because of the lack of results. Nowadays, Artificial Intelligence is back and has become one of the hottest topics as computers have evolved and are now able to run efficiently an AI. But the term is still wide and involves many aspects that will be further explained in this thesis.

To have a deeper understanding of the concept, he history of AI will be gone through. After that, machine learning and deep learning, which are both sub-parts of AI, will be explained as well as their differences. Lastly, the type of AI used will be highlighted and justified.

2.1.1 History

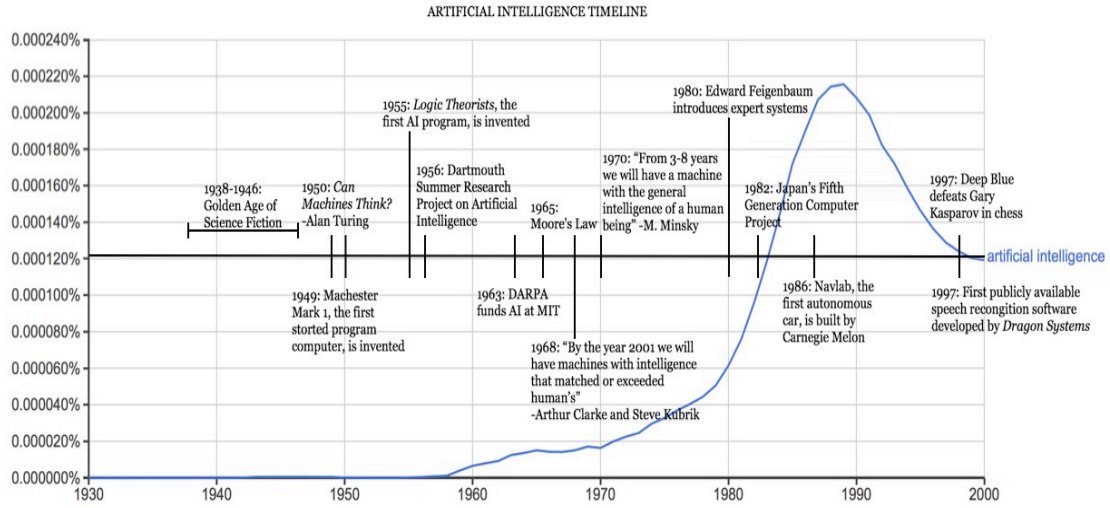

Figure 1 Artificial Intelligence history

At the very beginning, in 1950, Alan Turing, a mathematician, explored the mathematical possibilities of AI because, according to him, humans use available information and logic to solve problems and take decisions.

Computers were only able to execute command and were unable to store any command until the 1949 and thus, lacking a key feature essential to develop AI. To add to that, the cost of leasing a computer powerful enough ran up to 200’000 dollars per month and there was a need for a solid proof of concept and support.

Then, the race to AI started in 1956 during the Dartmouth College Artificial Intelligence conference hosted by John McCarthy and Marvin Minsky where the Logic Theorist was introduced. The proof of concept was a program that mimic the problem-solving skills of a human.

From 1957 to 1974, AI research was a success thanks to computers becoming capable to perform more tasks. Unfortunately, researchers stumbled upon a hurdle that jeopardized their work: computers couldn’t store enough information or process it fast enough. Because of that and the high expectations for AI, investors started reducing the funds and AI research fell into a new Ice age.

In the 1980’s, two main factors contributed to a new wave of AI research: an expansion of the algorithmic toolkit and a boost of funds.

Indeed, on the one hand, John Hopfield and David Rumelhart introduced a new concept named “deep learning”, discussed later in this thesis, and Edward Feigenbaum introduced expert systems that were able to mimic decision making processes.

On the other hand, the Japanese government invested approximately 400 million dollars from 1981 to 1990 with the purpose of revolutionizing computer processing, implement logic programming and improve AI with high results expectations. But many goals were not met and the funds disappeared, once-more.

However, thanks to the last huge funding of the Japanese government, engineers and scientists got inspired by the topic and that is why, even without the help of the government, many landmark goals of AI had been achieved. Gary Kasparov, reigning world chess champion and grand master got beaten by IBM’s Deep Blue in 1997 or Dragon System’s speech recognition software are examples of incredible advances in AI research.

Then, the age of Big Data appeared with the capacity to collect and store huge amount of data that can’t be processed by humans who have to rely on machine to do it. AI, one more time, is here to help, back to the big scene and still undergoing improvements. (Anyoha, 2017)

2.1.2 Machine Learning

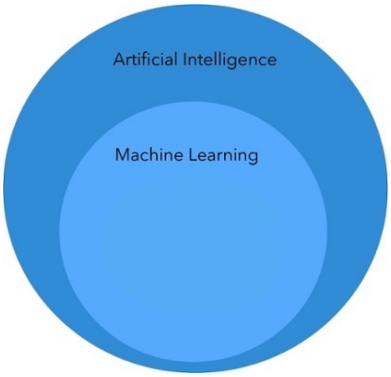

Figure 2 Artificial Intelligence and Machine Learning Structure[1]

According to Stanford University’s definition, machine learning is the science of getting computers to act without being explicitly programmed (Ng, 2018).

[1] https://www.datascience.com/hs-fs/hubfs/machine-and-deep-learning.jpg?width=418&name=machine-and-deep-learning.jpg

Machine learning is a way of creating an Artificial Intelligence, however, it is possible to have an AI without this method, although, it would imply writing millions of lines of code and the more complex the AI is, the more lines are needed.

The interesting part with machine learning is that the program considered in this thesis will be able to learn and adapt on its own without needing the help of Humans. In other words, it will be able to gain and evolve with experience just like a human being.

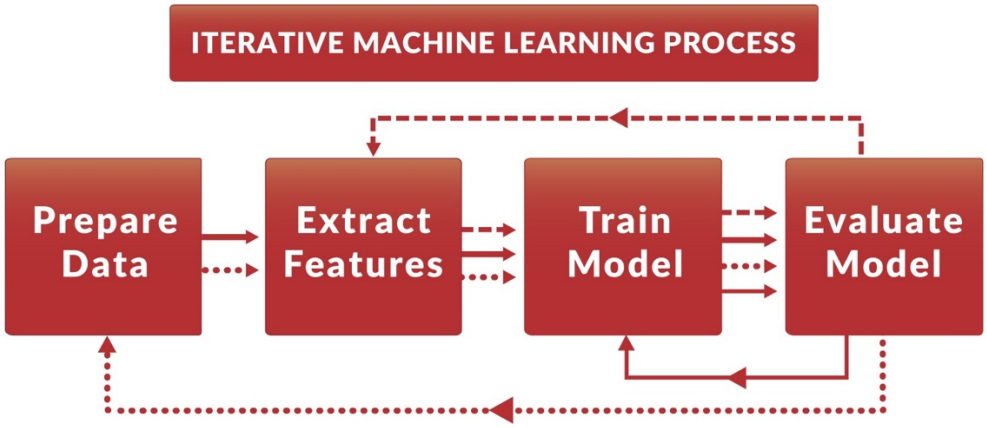

Figure 3 Machine Learning process[2]

In the case of machine learning, the AI will be trained with a dataset in order to gain experience and learn to perform the asked task. For example, in the object recognition field, an AI trained with the machine learning method will be given pictures of each object it has to recognize and be trained to learn how to recognize it. (McClelland, 2017 ; Ng, 2018 ; Pigeon, 2018.)

That way of implementing an Artificial Intelligence has been enable with the recent surge in processing power and thus, ability to work with a large amount of data. Indeed, the more data the program will be fed, the better and acurate it will be. To add to that, the nature of the data fed has to be consistent and relevent in order to have an AI performing the desired way. The importance of the dataset is hereby highlighted and with that, the fact that the human factor in the success or failure of such a program still remains important.

[2] https://cdn-images-1.medium.com/max/1600/1*WjXHRFcFT--7jPRWJ9Q5Ww.jpeg

To sum up, developers have come up with a new way of implementing an Artificial Intelligence, able to learn by itself. But what can be the involved improvements? One of the involved improvements is the so called “Deep Learning”.

2.2 Deep Learning

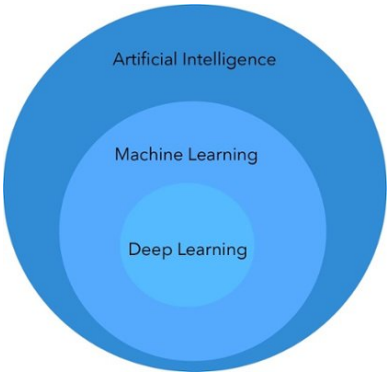

Figure 4 Artificial Intelligence, Machine Learning and Deep Learning structure

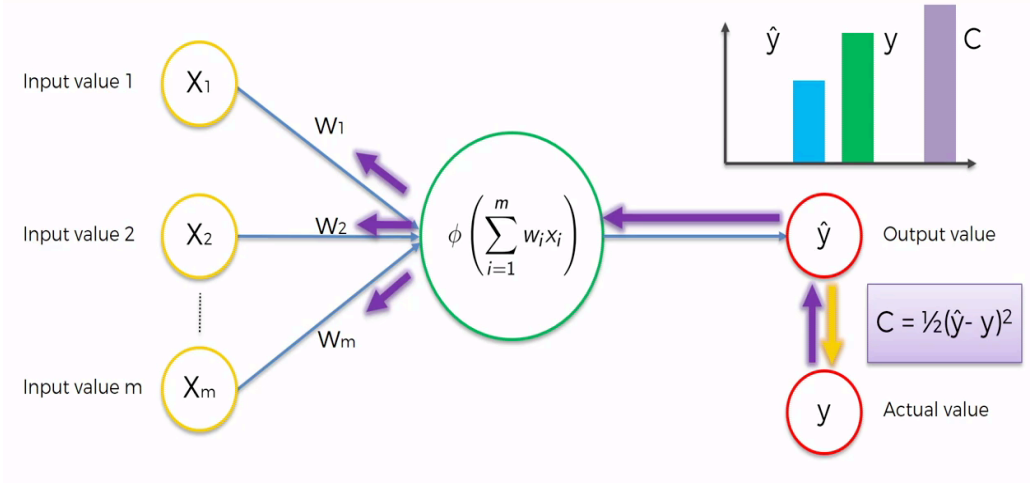

Figure 5 Deep Learning process[3]

Deep learning is a subfield of machine learning concerned with algorithms inspired by the structure and function of a brain called Artificial Neuron Network (ANN) and is the state-of-the-art method to implement machine learning. Indeed, it sees the Artificial Intelligence as a human brain that is composed of neurons and synapses that work together to resolve problems and learn from experience. Because of that, an Artificial Intelligence implemented with Deep Learning is referenced as an Artificial Neural Network (ANN).

To better understand ANN, an example is presented. As humans, we perceive our environment through different channels, different inputs -earing, sight, touch-. Based on those

[3] Picture taken from a course in Udemy

inputs, we are able to make deductions. An ANN will do the same as shown on figure 5. Inputs will be given to the machine, those will be weighted, because each input has more or less importance in the final deduction and given to a neuron that can be seen as a black box. It also called the hidden layer and is responsible for working on the different input values. The work consists of summing up the weighted inputs and give the result to an Activation Function.

2.2.1 The Activation Function

That function is here to determine if the result of the neuron will be processed further or not. This process is based on different algorithms whose most known ones are the threshold function, the sigmoid, the rectifier or the hyperbolic tangent. Every activation function has its pros and cons and influence. Because of that, the choice of which one to use relies on the nature of the input values but also on the output value.

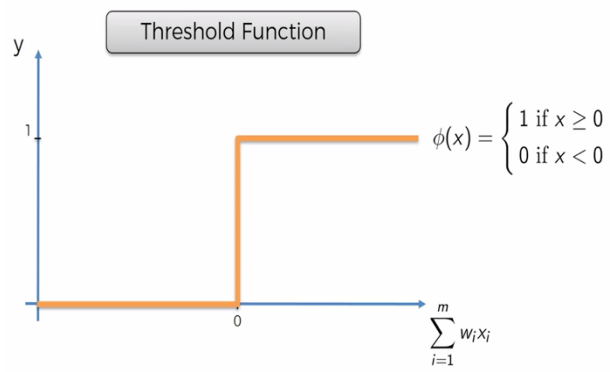

Figure 6 Threshold Function with a bias of 0

The x-axis represents the sum value of the weighted inputs and the y-axis, the result of the threshold function. The function will always return either 0 or 1 which is kind of like a Boolean that will be either true or false. The threshold function has a bias that is the same as a minimum required to be considered true. The bias value is decided by the developer to fit the needs. For instance, the threshold function will always return 0 as long as the sum of the weighted inputs plus the bias is less than 0 and 1 if it is equal or greater than 0. It means that the neuron will be taken into consideration if its sum plus bias is equal or greater than 0. The method is really straightforward, rigid.

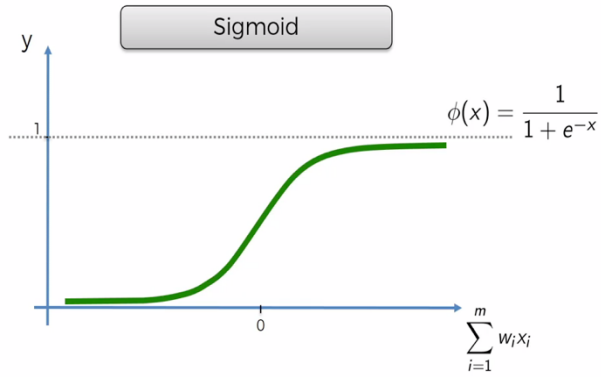

Figure 7 Sigmoid Function

The Sigmoid function is smooth compared to the threshold function and is especially useful in the output layer to predict probabilities. Anything below is dropped off and above will approximate 1.

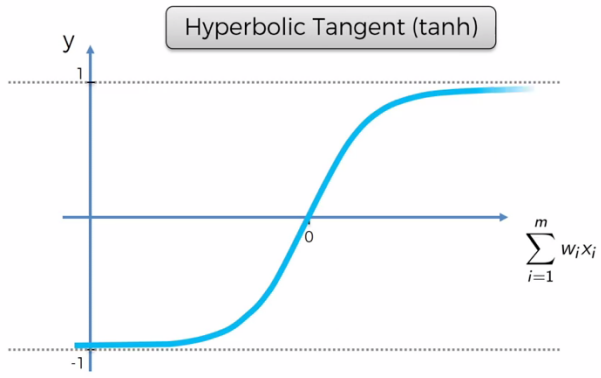

Figure 8 Hyperbolic Tangent Function

The Hyperbolic Tangent function is similar to the sigmoid function but the function goes below 0.

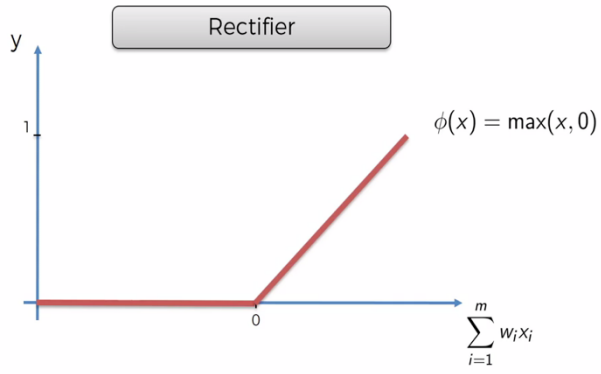

Figure 9 Rectifier Function

Even though it has a kink, the rectifier is one of the most popular activation function. Until the sum is equal to 0, the result is 0 but then, it increases gradually as the input value increases as well.

After going through the Activation function, the result is either given to another hidden layer or to the output layer. On the one hand, the output values can be single. For instance, in the case of estimating the price of an apartment, the output value will be continuous because it will be a price. On the other hand, it can be multiple. Here, most of the time, the output values will be composed of probabilities. For example, if the Artificial Intelligence has to detect the different fruits in an image, in the output layer, multiple probabilities of a fruit being in the picture will be received (Nielsen, chap1.html, 2017).

2.2.2 Backpropagation

The last step of Deep Learning is the backpropagation. The goal of this step is to understand how changing the different weights and biases in a network influence the Cost function. The machine will generate an output and compare it to the expected value with a Cost function. And the goal is to minimize the result. In order to do that, the ANN will adjust all the weights from right to left according to how much they are responsible for the error. The back-propagation can be done after each observation (Reinforcement Learning) or after a batch of observations (Batch Learning). The choice depends on the quantity of data used. Indeed, the more data available, the less the frequency of back-propagation will be.

$$C(w,b) \equiv \frac{1}{2n} \sum_x \|y(x) - a\|^2.$$

Figure 10 Cost function in its Quadratic form

As shown in Figure 10, one of the most used Cost Function is called the Quadratic Cost Function or mean squared error (MSE).

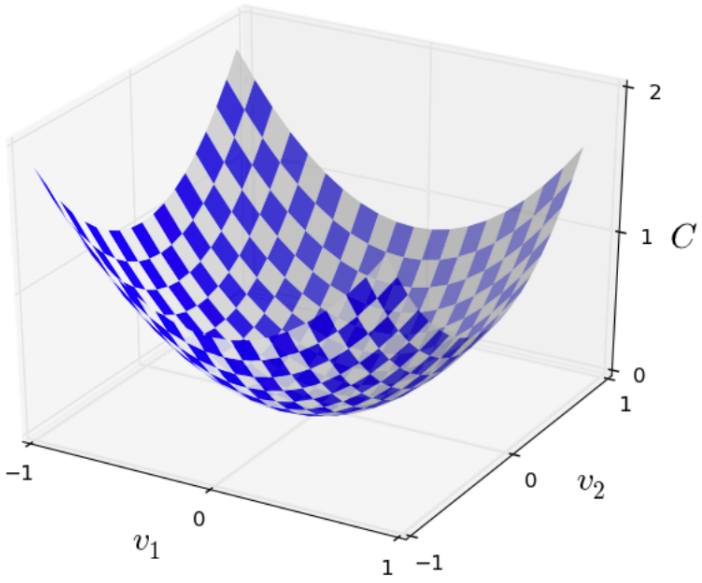

Figure 11 Quadratic Cost Function representation

The advantages of such a cost function is that, first of all, it never goes below zero as all the terms are greater or equal to zero. To add to that, it helps to determine the way of correcting the different weights and biases by aiming to minimize that function. Indeed, it can be extremely difficult to find out how to influence the weights and biases by trying to maximize the rightness in the training data. However, a smooth function represented in Figure 11 as the MSE is much easier to manipulate and understand.

Now, going back to minimizing the Cost function, in a graphical representation as in Figure 11, the first idea to determine the minimum would be to use calculus and try to find where the derivative would be equal to 0. Unfortunately, it would only work for simple Cost function using only two variables and the more complex a Neural Network will be, the more variables will be included in the Cost function. Because of that, a new way of determining the minimum is proposed. The idea is to imagine the function as a valley where we will let a ball roll down the slope until it finishes at the bottom. What's interesting to know is that the behavior of the ball corresponds to a partial derivative of the Cost function that will give the direction; if it is going uphill or downhill. From here, the goal is to minimize the Cost until a global minimum is found as well as a partial derivative as close from zero as possible (Nielsen, chap2.html, 2017).

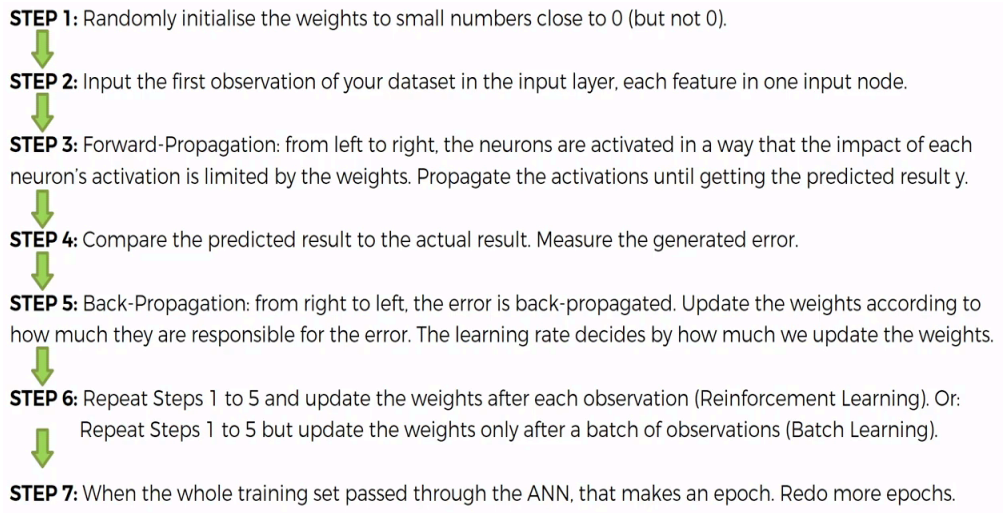

Figure 12 The 7 Deep Learning Steps

All the steps involved in Deep Learning are part of an iteration or epoch. And the AI will run as many epochs as needed in order to minimize the Cost function. (Berkman, 2017 ;Brownlee, 2016 ;Dettmers, 2015 ;Keenan, 2018 ; McClelland, 2017 ; Moujahid, 2016.)

After the training, the Artificial Intelligence can be used to predict a behaviour. For instance, if we take the same example of the house pricing, given the right input values, the algorithm will be able to answer with an estimated price.

To add to that, having multiple outputs is quite common and, we need to choose the best output. In order to take the right decision, the output values are given to another funtion called the SoftMax function that will iterate through them and select, for example, the ouptput with the highest probability.

2.2.3 Digging deeper in how a Neural Network learn

As humans, making huge mistakes help us to learn faster and the harder we fall, the more we learn. So, if the same concept is applied to machine learning, setting weights and biases badly wrong should help the algorithm to learn faster. As it is not that is not the case, in order to understand the reason behind the phenomenon, remember that neurons learn by adjusting the weights and biases they are related to. In other words, saying that a neural network is “learning slowly” is equivalent to saying that the partial derivatives of the Cost function are small.

To improve the learning capacity of a neural network, some key-elements can be influenced but we will focus on the Cost Function.

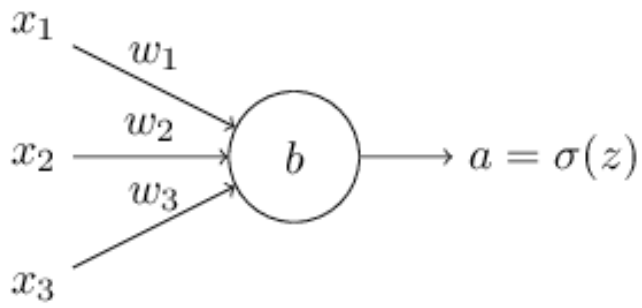

Figure 13 Neuron receiving three input values

In figure 13, a neuron receives three input values x_1, x_2, x_3 with their corresponding weights. Then, the output value a will be the sigmoid of the sum of the weighted values plus the bias (z).

$$C = -\frac{1}{n}\sum_x [y \ln a + (1-y)\ln(1-a)]$$

Figure 14 Cross-Entropy cost function for a neuron with three input values

In the case shown in Figure 13, the Cross-Entropy cost function equation will be as shown in Figure 14. While the reason why that function can be considered a cost function is not visible at a glance, two properties in particular make it so:

- the function is non-negative
- and if the output value is similar to the desired one, then the function will be close to 0.

The main difference between the Cross-Entropy and the MSE is that the first one resolves the problem of learning slowing down.

$$\text{Step 1: } \frac{\partial C}{\partial w_j} = -\frac{1}{n}\sum_x \left(\frac{y}{\sigma(z)} - \frac{(1-y)}{1-\sigma(z)}\right)\frac{\partial \sigma}{\partial w_j} = -\frac{1}{n}\sum_x \left(\frac{y}{\sigma(z)} - \frac{(1-y)}{1-\sigma(z)}\right)\sigma'(z)x_j$$

$$\text{Step 2: } \frac{\partial C}{\partial w_j} = \frac{1}{n}\sum_x \frac{\sigma'(z)x_j}{\sigma(z)(1-\sigma(z))}(\sigma(z) - y)$$

$$\text{Step 3: } \frac{\partial C}{\partial w_j} = \frac{1}{n}\sum_x x_j(\sigma(z) - y)$$

Figure 15 Derivative process to understand the Cross-Entropy function

In the last step of Figure 15, the derivative of the Cross-Entropy cost function by the weights of the actual neuron tells us that the rate at which the weight learns is controlled

by $(\sigma(z) - y)$ which corresponds to the error in the output. Because of that, the bigger the error, the faster the neuron will learn. That is the opposite of the Quadratic Cost Function (Nielsen, chap3.html, 2017).

2.3 Component Recognition

Before even thinking about generating code, it is necessary to understand how the different components of an app are going to be recognized.

As humans, we have the capacity to differentiate objects, animals, colors, seeming easy for us. But how does our recognition mechanism work? With experience, our brain is able to store and label objects; in other word, it is able to classify them. Thanks to this ability, we are able to react accordingly in a fight-or-flight situation for example because we know that depending on the situation, it is common sense to defend ourselves or just go away as fast as possible. So, to recognize the different components of a design, the computer needs to be able to do the same. It has to train itself to know what a header, a footer, a paragraph, images…

For the machine to be able to do the processing, a Convolutional Neural Network (CNN) will be used. After, we will go through how the image recognition works and step by step, a deeper introduction into the subject until we define what a CNN is.

2.3.1 Image processing

Let's start with understanding how an Artificial Intelligence can recognize handwritten numbers. From the point of view of a computer, an image is nothing else than an array of numbers with each number representing the gradient of black. Because of that, to train the neural network, we only need to feed it arrays of numbers representing, for example, the number 8.

Through image processing, the output will not be unique anymore. In fact, two outputs will be received; those being the probability of "the image has an 8" and "the image does not have an 8" if we take the number 8 as an example again.

Figure 16 Output of image processing of a handwritten 8

However, the outcome of a processed image strongly depends on the dataset used to train the neural network. Indeed, if trained with images of handwritten numbers where the number is perfectly centred every time, the machine will not be able to recognize a given number if it is not positioned the same way. And here is when we reach the limits of image processing with a normal fully-connected neural network.

The problem resides in the fact that the output of the neural network when processing the image of a number, in that case, depends on the structure of the dataset.

Figure 17 Output of image processing with centred number in dataset

In order to resolve it, the first solution that comes to mind would be give more data to the machine and train it more because the more data it will be fed, the more accurate it will result. In theory, it can resolve the problem. But then, the dataset will need to cover all the many possibilities that the item that has to be recognized can be represented in an image. And that shows the limits of the solution. Indeed, it would be impossible to train the neural network with a dataset that fully covers those possibilities depending on what has to be recognized.

Whereas it is not optimal, the deep neural network that comes out of that solution is interesting. Because of the increased data, the problem becomes harder for the machine to resolve, thus, more layers would have to be added to the neural network in order to make it more effective and able to understand more complex patterns.

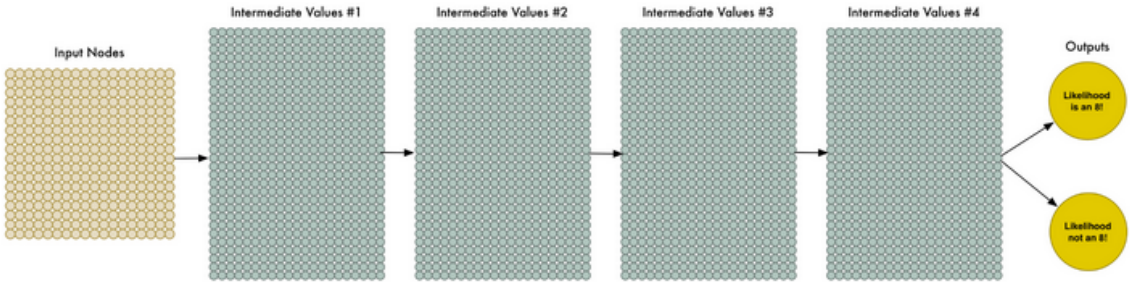

Figure 18 Deep Neural Network

As written by Adam Geitgey: “We call this a “deep neural network” because it has more layers than a traditional neural network”. Another possibility demonstrated by Adam Geitgey is “to scan all around the image” until the machine is able to resolve the problem. It is called the Sliding Window (Machine Learning is Fun! Part 3: Deep Learning and Convolutional Neural Networks, 2016).

2.3.2 Convolutional Neural Network

The human brain is able to recognize an object whatever the environment because it doesn’t need to re-learn for every situation. Indeed, it is able to memorize the specific properties related to an object and use them to tell us if it is present or not.

And that is exactly what the machine needs to be able to do. As said by Adam Geitgey: “we need to give our neural network understanding of translation invariance”. (Machine Learning is Fun! Part 3: Deep Learning and Convolutional Neural Networks, 2016)

In order to give the machine this ability and thus, to improve its skills at recognizing an item in an image without needing to feed it with data that cover every situation, the Convolutional Neural Network appears. This technique to implement machine learning is specific to image processing and is composed of multiple steps: breaking the image, feed the image, store the result, downsampling, predict.

Figure 19 Step 1 – Breaking down of an image

The first step of breaking the image consists in cutting the image in smaller, overlapping images of equal sizes. In the figure 13, an image has been broken down in 77 overlapping tiny images of equal size.

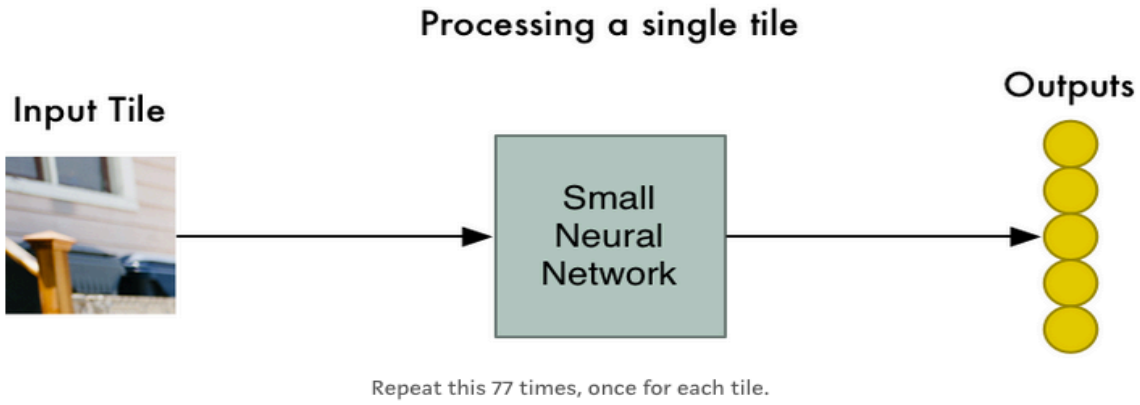

Figure 20 Step 2 – Feeding of a tiny image to a small neural network

For the second step, all the tiny images are fed to a small neural network that is going to analyze it and see if the expected item is present or not.

But here, there will not be any backpropagation because all the tiny images are going to be fed to the neural network with the same weights. Keeping the same environment between images helps to process them all the same way.

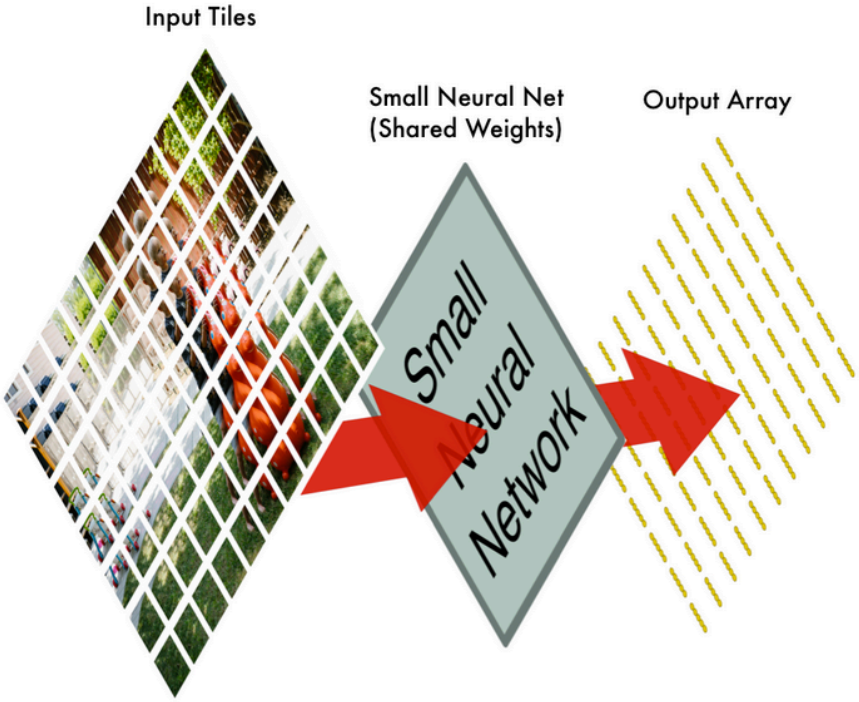

Figure 21 Step 3 – From an array of images to an array of results after processing

In figure 15, the results of each image fed to the neural network are stored in a new array. Because of that, the arrangement of the original tiles is kept. It is important to keep the arrangement of the tiles like the original image because we want our machine to think the same way as humans. The closer two individual images are, the more related to each other they will be. If we shuffle the order at this point, the correlation between the relative position and the relation between pictures would be lost.

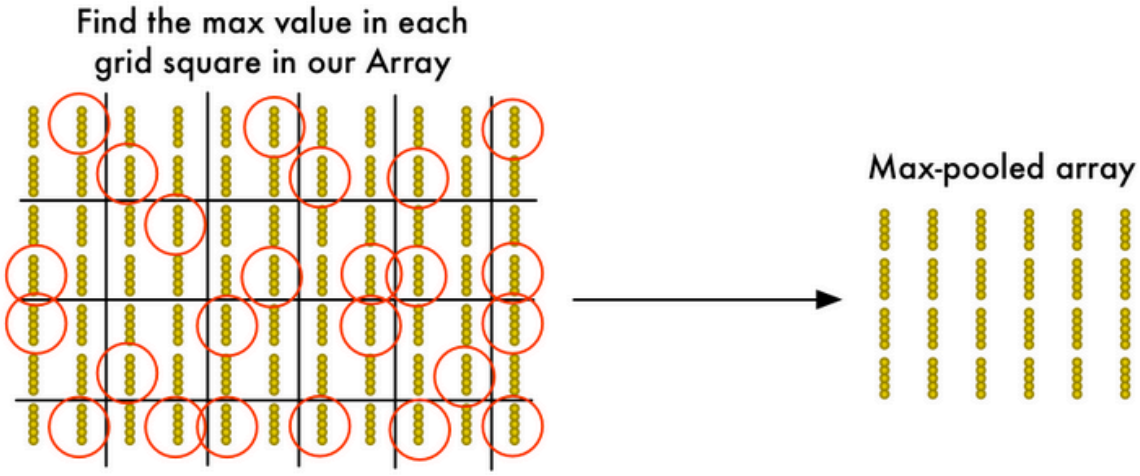

Figure 22 Step 4 – Downsampling

The figure 16 illustrates the concept of downsampling. Because the order has been kept, the array is divided in 2x2 grid squares and from those small parts, only the most interesting output, being the one with the best similarity to the expected output, is kept.

This process helps the machine to reduces the size of the array that it has to process while keeping the important part.

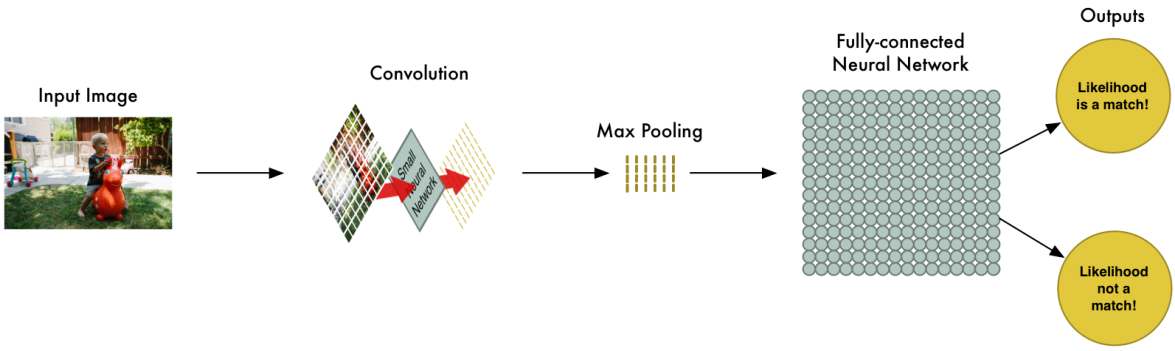

Figure 23 Step 5 – Full Convolutional Neural Network

The fifth and last step consists of feeding the reduced array to a fully-connected Neural Network which will be responsible of processing it like any image only composed of numbers for the computer and give the final output.

All those steps are key factors of a Convolutional Neural Network aimed at processing images. A simple one will only iterate through the step 1 to 4 once but more complex ones can be composed of multiple iterations where each small neural network will be responsible of processing the array in order to look for a specific detail and thus, reducing the array to the very most important parts. And after being fed to the fully-connected neural network, it will give us the most precise output. (Dettmers, 2015 ; Geitgey, 2014 ; Keenan, 2018 ; McClelland, 2017 ; Seif, 2018 ; gk_, 2017.)

2.4 Current situation

With the Artificial Intelligence coming back to the big stage, AI researchers are exploring how this technology can be used to help developers bring new powerful tools. As a matter of fact, I am not the first one to research how to generate code from mock-ups. Big companies, Airbnb for example, have already come up with experimental solutions related to the topic discussed in this thesis. Not being the first one to study the ways of using the Artificial Intelligence, I therefore base my analysis on two main researchs previously proposed by Airbnb (2018) and tony belltramelli (2017), which strongly relate to mine.

2.4.1 Airbnb's sketch2Code

Airbnb is widely-known for their renting services but is betting big on automation and believes that the process to transform design into code will, in the future, be done by an Artificial Intelligence.

The guideline behind Benjamin Wilkins's work is based on the idea that "The time required to test an idea should be zero" (Generating code from low fidelity wireframes).

Airbnb has come up with a solution called sketch2code. Before implementing an application, the first stage of every project is: sketching. This part of the process is extremely important to create a base for the final product and how it will look like. It is also the moment where the design complexity is set.

But, how can a designer test the feasibility of the design? How can he/she know that it can be implemented? Right now, the only solution is to write the corresponding code, face the problems and reiterate by prototyping and coming back to the generation phase of ideas until design and feasibility blend together. That is why sketch2Code is relevant and Benjamin Wilkins thought that "Sketching seemed like the natural place to start" (AirBnb, 2018).

So, starting from the root of the problem, sketch2Code is able to scan the mockups made by the designers and translate it into code. The capacity of an Artificial Intelligence to generate code in the blink of an eye will help designers to test their design in real-time and help the developers to improve the generated code. (AirBnb, 2018)

2.4.2 Tony Beltramelli's pix2code

After Airbnb, another actor explored the topic. That person is Tony Beltramelli, Co-Founder if Ulzard Technologies. He researched how to automate writing code to implement the User Experience part of an app, led by the same reasons as me. In fact, he says:

> The process of implementing client-side software based on a Graphical User Interface (GUI) mockup created by a designer is the responsibility of developers. Implementing GUI code is, however, time-consuming and prevent developers from dedicating the majority of their time implementing the actual functionality and logic of the software they are building. (Beltramelli, 2017)

In his case, the Artificial Intelligence will analyze the structure of a screenshot to determine the components and then translate it into code. The difference with Airbnb's solution resides in the programming language used to generate the code. Indeed, while Airbnb has

focused his solution around React, pix2code is able to produces code either for iOS, Android or the web.

2.5 Normalized architecture

Now that the technical requirements have been explained, it is time to understand the purpose of the research.

One of the biggest problem in an IT project is the learning curve of a newcomer when he or she joins a team. Before even starting to write a line of code or actually getting involved in the project, that person needs to understand the state of the project namely, the technology, the architecture, the conventions used. Depending on the stage, the complexity of each step can vary a lot.

As Artificial Intelligence remains just a machine, the generated code will always be the same as long as the same technology and programming language are used. Because of this ability to continuously produce the same output, it is possible to normalize the written code between projects. For that reason, the learning curve will be improved. Any newcomer will already know how the code is structured and where to look for in order to fix a bug.

Moreover, through generating code with an AI, it is possible to keep developing it to always use the latest state of the art libraries, plugins and more, without needing to spend as much time as a human would need to learn it. The IT field is ever changing and depending on the sub-field, those changes can happen at a frightening speed resulting in developers being unable to follow the rhythm because of the surreal time they would need to invest in order to master the new technology.

So, the machine could be an important asset to a software development company by providing the same code all the time and helping employees to focus on other tasks rather than having developers implementing the design of the software over and over again.

To better understand the improvements in the architecture that a machine can bring, we will look deeper into the Object-Oriented language like Java. In this case, one of the key-factor resides in the low coupling and high cohesion of the classes. It means that classes related to the same task should be together and limited in their relationships with those doing another one. The point here is to avoid any global impact to an update in a class, differentiate the roles of every package of components and know exactly where to look for

bugs. Therefore, a developer needs to pay attention to how to structure the code to respect the principle.

In order to do so, an important amount of time has to be invested in to researching and trying to put together the architecture. With experience, it gets better and faster but, in some cases, it could occur that developers stick to their habits and stop improving the model because it is working fine as it is. However, if fed enough projects using the low coupling and high cohesion principle, the Artificial Intelligence is able to recognize a pattern and pick the best of every project in order to find the best architecture.

To illustrate the pattern finding ability of an AI, Adam Geitgey (2016) explains how it is possible to generate Super Mario Bros maps after translating each component of it in a symbol recognizable by a computer.

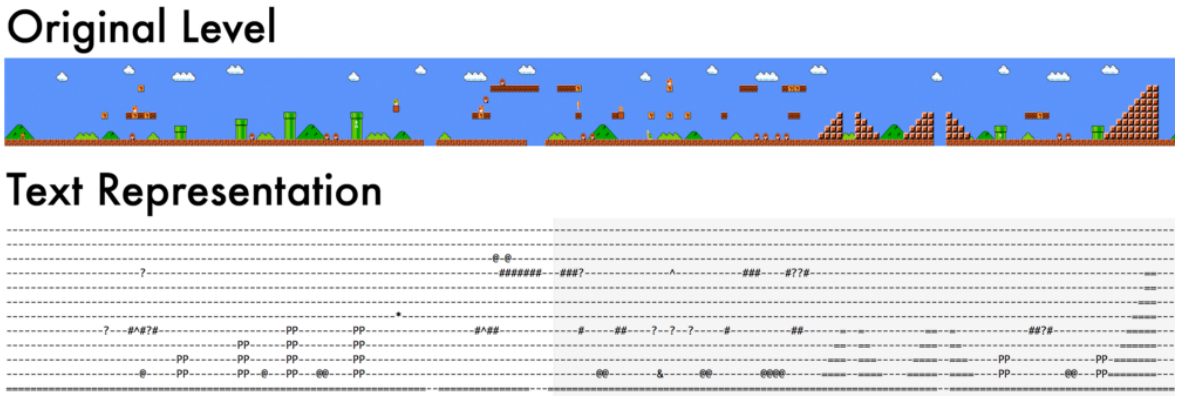

Figure 24 Super Mario Bros map translated for computer (Adam Geitgey, Machine Learning is Fun! Part 2: Using Machine Learning to generate Super Mario Maker levels, 2016)

In the figure 19, the translation is done so that :

- "-" is a blank space
- "=" is a solid block
- "#" is a breakable block
- "?" is a coin
- "P" is a pipe
- "@" is an enemy

… and so on with every character representing a component of the map.

After training the Artificial Intelligence with the map, it is able to understand the intrinsic properties of the map and thus, notice many aspects like

- each column as to have the same height
- a pipe needs to reside on top of two bricks wherever they are
- no obstacle should obstruct the player to advance further. (Geitgey, 2016)

2.6 Tools used

In order to implement a solution and create an Artificial Intelligence able to do the expected work in this thesis, different tools are needed: a programming language and an Integrated Development Environment.

Nowadays, Python is the most used programming language in the machine learning field with 57% of data scientists and developers using it. From that percentage, 33% prioritize it to start a project. One of the main reasons for that choice can be found in the wide range of frameworks which brings tools to developers like TensorFlow or PyTorch. (Voskoglou, 2017) This is the reason motivating me to choose Python as the language to build the proof of concept joined.

Then, to code, an Integrated Development Environment is needed. It is a software that facilitate application development. In general, an IDE is a graphical user interface (GUI)-based workbench designed to aid a developer in building software applications with an integrated environment combined with all the required tools at hand. Based on experience using IntelliJ from Jetbrains, I decided to use their Python IDE named PyCharm in order to help me to create my solution. (Techopedia, 2018 ; Vasconcellos, 2017)

To go deeper in the machine learning aspect, TensorFlow will be the framework used to create the different parts needed to make my project.

> It is an open source software library for high performance numerical computation. Its flexible architecture allows easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices. Originally developed by researchers and engineers from the Google Brain team within Google's AI organization, it comes with strong support for machine learning and deep learning and the flexible numerical computation core is used across many other scientific domains. (TensorFlow, 2018)

A lot of companies like AirBnb or NVIDIA are also using that framework and it is reassuring to know that other big leading companies trust Google with that tool to help them develop their own machine learning solution.

2.7 Consequences in the IT field

Generating code automatically with the help of an Artificial Intelligence hasn't been implemented well enough to be used all the time. Because of that, there is no study regarding the possible consequences on employment.

However, history has shown us that every big change comes with its advantages and disadvantages. My assumptions are the negative impact on the IT fields will be eclipsed by the good ones. Indeed, even though the developers responsible of implementing the design are going to lose some ground to the machine, they will, at the same time, have more time to develop other skills that will further improve the field.

For example, on the one hand, Artificial Intelligence awakening leads to an increase in ability to manipulate complex and numerous data. On the other hand, those same data come from users through all the different available services. So, in a more and more connected world, how does one's privacy can remain private? How can we protect our integrity? In my opinion and based on the interview of Zvika Krieger, co-leader of the World Economic Forum's Center for the Fourth Industrial Revolution, by Nick Johnsone, security is one of the fields that is going to be affected because, as written, "We don't want technology to take away people's freedom; we want it to give people more choice and more autonomy". (World Economic Forum, 2018 ; RichmondVale, 2016)

Even though those big changes are happening, the World Economic Forum's Center for the Fourth Industrial Revolution is helping to supervise them and, by cooperating with companies and people from different horizons, develop policy frameworks and that will accelerate the benefits of science and Technology.

3 Empirical Part

Now that the theoretical part and knowledge has been established, the implementation of a working solution can start. In order to provide a proof of concept to the thesis, an Artificial Intelligence able to generate code based on a simplistic design mockup will be created from scratch. The simplistic part resides in the fact that the solution will only focus on recognizing images and text as well as the layout relative to each component.

The extent of the ability of the algorithm to detect and generate code will be narrowed down as I lack time, sources and the knowledge to implement a perfectly working solution able to cover all the different components that can be present in a HTML page.

In other words, the objective is to implement an AI able to detect simplistic representation of paragraphs and images and translate them to code.

	Task Name	Duration	Start
1	Complete project execution	21 days	16.04.18
2	Component Recognition	7 days	16.04.18
3	Understand pre-trained models	3 day	16.04.18
4	Duplicate relevant code	2 days	19.04.18
5	Modify code to take own dataset	1 day	21.04.18
6	Testing	1 days	22.04.18
7	Layout Recognition	7 days	23.04.18
8	Understand how to get position	2 days	23.04.18
9	Update code to store position	3 days	25.04.18
10	Testing	2 days	28.04.18
11	Code Generation	7 days	30.04.18
12	Understand pattern recognition	7 days	30.04.18
13	Modify code to fit code pattern	4 days	02.05.18
14	Introduce code generation with parameters	13 days	04.05.18
15	Testing	1 day	06.05.18
16			

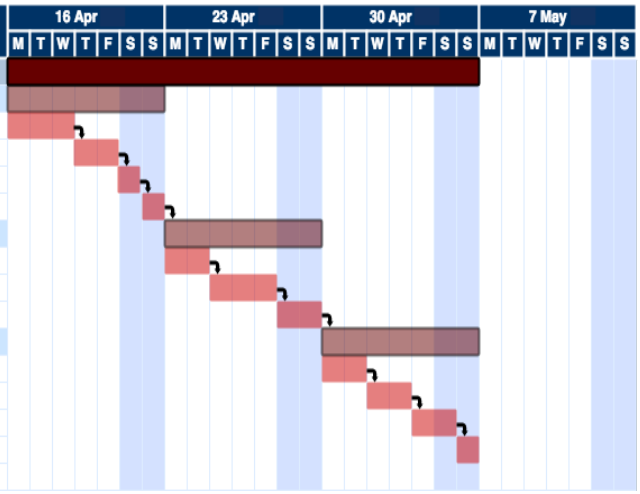

Figure 25 Implementation plan

In figure 25, the different steps of development are shown. The main part of my solution will be the component recognition. The importance of this component and its difficulty of implementation is the reason I decided to approach this element first. Then, it will be updated in order to store and remember the location of the recognized components through the layout recognition. Lastly, with all the cards in our hands, it is my aim to generate a precise, not random code, created under parameters representing the different elements found in the first part of the algorithm.

3.1 Dataset

Trying to implement an Artificial Intelligence able to generate code like that is something completely new. As a matter of fact, no dataset complete enough actually exists that could

help to sufficiently train an AI. A dataset is crucial to machine learning because the more data, the more accurate the algorithm will be.

For the solution, as I needed to produce the dataset by myself, I made use of the many resources available to me and collected the material needed to support this thesis of the different components there with a focus on texts and images. The task was really long and laborious. After revising all the material, I realized that the algorithm wasn't good enough and only worked on a few and specific test images. That problem led me to decide to choose two types of elements because they represent the main components of a website and they seemed like the most easily recognizable parts. In order to create a dataset big enough to train and evaluate my AI, I drew two shapes: a black square with a cross in the middle and a black square with horizontal lines. The first one represents an image and the second one a paragraph. Now that the simplistic representation has been created, I converted them to black and white pictures. The same picture will be fed to a loop that will iterate through all the integers from 1 to 359 and save a new picture of the simplistic representation rotated by one degree each time. Thanks to that solution, the dataset for each component went up to 360 pictures. An example of a paragraph representation after a rotation of 0 degree (original) and 10 degrees can be seen hereafter in Figure 26.

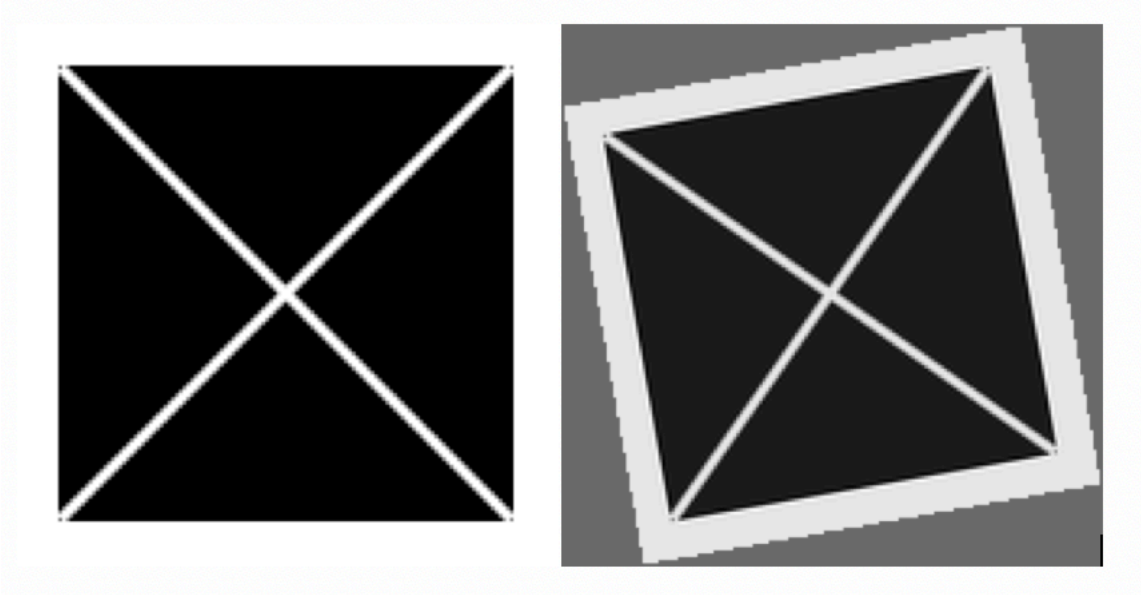

Figure 26 Picture representation at 0° and 10°

Another option possible option for increasing the size of the dataset would be to blur the pictures, change the contrast, the brightness and many other factors. However, a simpler and more concrete representation was chosen.

```
def main(unused_argv):
    picture = ndimage.imread("picture.png", flatten=True)
    paragraph = ndimage.imread("paragraph.png", flatten=True)
    picture_rot_array = []
    picture_array_labels = []
    paragraph_rot_array = []
    paragraph_array_labels = []

    for x in range(0, 360):
        picture_rot = ndimage.rotate(
            picture,
            x,
            mode='constant',
            cval=100
        )
        picture_rot_array.append(picture_rot)
        picture_array_labels.append("picture")
        mpimg.imsave(
            "data/picture/train/picture{}.png".format(x),
            picture_rot,
            cmap=plt.cm.gray
        )

        paragraph_rot = ndimage.rotate(
            paragraph,
            x,
            mode='constant',
            cval=100
        )
        paragraph_rot_array.append(paragraph_rot)
        paragraph_array_labels.append("paragraph")
        mpimg.imsave(
            "data/paragraph/train/paragraph{}.png".format(x),
            paragraph_rot,
            cmap=plt.cm.gray
        )

    print("Data Set for pictures created !")
```

In the code here before, numpy has been used to work on the image. The .imread() function will find the picture at the given location and convert the image into a numpy array which is the representation of the image in vectors.

Then, the for loop will iterate from 0 to 360, with 360 excluded, and rotate the loaded picture by rearranging the vectors through the .rotate() function. After that step, the .imsave() function save the pictures to the given path.

All those steps are repeated 360 times in order to create my data. Being a proof of concept, I wanted to demonstrate that it was possible to detect those shapes and convert them to code. That dataset has been used to aim to train the prototype of AI I present.

3.2 Component Recognition

The Component Recognition is the most important part of the whole process. Indeed, it has to be thoroughly trained in order to recognize all the different elements presented in a design mock-up.

It was my goal to reuse the code from TensorFlow training set and modify it in order for it to recognize my own pictures but unexpectedly, this idea faced the image manipulation problem.

I then implemented an example using Watson API that returns to me probabilities regarding the appearance of an element. In a way, recognizing an element seems rather easy but the number of possibilities a design can present is based on the designer imagination; thus, they are near infinite… That's why, the capacity of self-learning of an algorithm is primordial.

With that problem came the question of how to distinguish a component, what are its intrinsic characteristics? It is rather complicated and complex to define a component by mere criteria. For instance, an image can have different shapes, different border size or colour. The specific elements of an image were narrowed down and I then arrived to the conclusion that as for now, it is limited to a black square with a white X in the middle. It is a rather simplistic representation of an image but for a proof of concept, that would be enough. As for a paragraph, I decided to limit the representation to a black square with horizontal white stripes.

The biggest challenge here resides in the limited possibilities of writing a paragraph and inserting an image. Design is an art and therefore, the imagination is the only limit. Based on my knowledge and the current sources available to me, I was unable to find a pattern in the case presented.

By using the code found on Tensorflow tutorials, I am able to retrain a model in order to recognize the two components I have created with great precision. However, that precision is high only because of the simplistic representation of an image and a component. In reality, as stated before, those elements can be found in many shapes and aspects thus, making the recognition process much more difficult.

The retrain.py code has been modified in order to take as default parameters the right paths and directories specific to my project. Because of that, by running the following command, it will take in the data set created before and train the model. See Appendix 1 for the whole code.

To go deeper into the code, the main methods to understand are the following:

- main(_),
- prepare_file_system(),
- create_image_lists(image_dir, testing_percentage, validation_percentage),
- create_module_graph(module_spec),
- add_final_retrain_ops(),
- add_jpeg_encoding(module_spec),
- cache_bottlenecks(),
- add_evalutation_step(result_tensor, ground_truth_tensor),
- get_random_cached_bottlenecks(),

The main(_) function is the starting point of the retraining. It initializes all the different global variables and calls the methods needed to train and evaluate the model. That method contains all the steps needed to have a model able to recognize the components. Another responsibility of main(_) is to detect errors as soon as possible and stop the process in order to avoid unnecessary work.

3.2.1 Environment setup

First of all, the environment has to be set up in order to train the model. Indeed, what is going to happen is that a pre-trained model will be used and on top of that, a new output layer is going to be added. That last layer is the one that will be able to recognize our components.

Different files are needed during the retraining and saved in directories. prepare_file_system() is here to create those directories in case they don't exist yet. Its task is quite simple but important as it must know where each created file is stored.

In create_image_lists(image_dir, testing_percentage, validation_percentage), the different lists of images are produced. It is its task to check if at least 20 images are presented in the subfolders; less than that wouldn't be enough to properly train the AI. The returned result is an array where each key is the name of the subfolder aka the label of the component, here it will be either "paragraph" or "picture" and each cell will contain the path to the subfolder, the training images, the testing images and the validation images. It is responsible of splitting all the images into three types of usage for the model to train, test and validate the process.

Then, the graph for the model is created through the create_module_graph(module_spec) method. It receives the specifications from the hub module of Tensorflow as an argument and generate a pre-trained graph that is going to be used to train with the images provided by the developer. Here, pre-trained model able to recognize pictures is used in order to speed up the process. Otherwise, the AI has to learn from the beginning how to recognize a picture.

After that, a new layer is added on top of the pre-trained graph with the add_final_retrain_ops() method. That layer will be the one used to recognize the picture and paragraph element. It returns the Tensor for the training and cross-entropy results as well as tensors for the bottleneck input and ground truth input. The last one contains true values of the expected output.

The data set generated before contains picture in jpeg format. However, that specific format needs to have a pre-processing in order to be used in our model. That's why the add_jpeg_decoding(module_spec) method is used, it adds operations to perform JPEG decoding and resize the picture for the graph.

The next step consists of creating the bottlenecks or image feature vectors that are going to be used for the training. The process can take up to 30 minutes depending on the speed of the machine but once it is done, it will be re-used every time. A bottleneck contains information on an image. It is important to create them because in this case, each image can be used multiple times and converting an image into a vector representation takes time, so it is faster to create all of them beforehand.

3.2.2 Training

Now that the environment has been set, the training can start. In this section, the different training steps will be described and explained so the process, although complex, will result to be easy to understand.

First of all, we need to be able to evaluate the accuracy of the new layer that will recognize our components. The add_evaluation_step(final_tensor, ground_truth_input) method fulfills that task and inserts the needed operations. The received arguments are, first, the final_tensor which is basically the layer we are training and, secondly, the ground_truth_input that have true expected values to compare to the future output values received and return a tuple of evaluation step and prediction.

Then, we will iterate through the number of training steps requested on the command line with a default value of 4000. During each iteration, we are going to get_random_cached_bottlenecks().

With those bottlenecks received, a training step is run after which a summary is saved for the TensorBoard visualisation. When all the training steps have been completed, the model can be saved in a checkpoint. and we can go on with an evaluation step where the model is going to be tested so the ability to recognize the elements can be verified.

Another possibility would have been to use the Watson Visual Recognition API and feed it with pictures of paragraphs and pictures. However, for the same reason as stated before, the number of ways to represent those two elements is near infinite as it is based on one's imagination and therefore, this possibility was not feasible and was tossed away as even after collecting visual representations of the components of different websites, I was still unable to generalize "a picture" or "a paragraph" as concepts. The related code can be found on my GitHub as a proof of concept for my research (Cheng, 2018).

3.3 Code Generation

The idea behind the code generation is that based on the probabilities received from the component recognition layer, I would be able to generate the corresponding code. In that matter, it wouldn't be a random output but instead, a parametrized output where the result would take into consideration which element is really present or not in the design mock-up.

In order to implement that, I opted for using a Recurrent Neural Network that is extremely good at predicting words in a sentence based on the previous words used in the text and adapt it to learn the structure and philosophy behind HTML programming and especially the opening and closing of tags.

For that part, I found some re-usable code from Karpathy where he trains his model to predict the next words in a sentence. I updated it in order to use the source code from the welcome view of GitHub and trained it with that code as data set (karpathy, 2016).

Regarding the use of the outputted probabilities from the component recognition part to parametrize the code generation, unfortunately, I didn't reach any conclusive solution. The only way that would be possibly fulfil my expectations would be to create a switch and output the corresponding code based on the probability and thus, the minimum required for an element to be considered detected and recognized. However, that solution would be

time inefficient and resource consuming as it would require that, for all the different tags available in HTML, a case would be created. So, here-after, the explanation in depth of Karpathy can be found and my update to train the model with HTML code.

The model is not using Tensorflow as the component Recognition. Instead, it is based on Torch which is scientific computing framework with the wide support for machine learning algorithms that pus GPUs first. It is easy to use and efficient, thanks to an easy and fast scripting language, LuaJIT, and an underlying C/CUDA implementation (Torch, 2018).

Lua is sequential execution. By that, I mean that when the script is launched, it will go through it from top to bottom. Because of that, the methods that are going to be used later on have to be initialized before-hand. That's why the most important part of the code is at the bottom. That method will basically iterate through the number of trainings needed and call the methods responsible of the different steps of the training.

One part that starts line 112 loads the data from my input.txt where the source code I provided can be found and create the vocabulary database based on the it. After that step, the RNN is flattened to be used in a tensor. Then, the weights are initialized randomly with small uniform numbers.

The eval_split() function is responsible to evaluate the loss over an entire split, usually at the end of an iteration and the feval() function will manage the epoch and the backpropagation as well as returning the loss for the current batch that is evaluated.

To summarize, the proof of concept is composed of three main parts: the data set creator, the component recognition and the code generator. By working together, the related code should be created as an output and should be usable for improvements by developers or for usability testing regarding the User Experience for example.

Even though the implementation is not a working solution, it gives an idea of the difficulty of creating such a solution and experience on what was successful or not. It has been really interesting to work on the topic discussed as it required a lot of commitment and learning that was, in that case, lacking as machine learning is quite new for me. But, with a deeper understanding of the subject and the way it is possible to link those three parts together, improvements can be achieved in the future.

4 Discussion

After going through the implementation part, the solution is, unfortunately, not a working one. If we consider each part separately, some good points can be highlighted. First, regarding the dataset creator, even though it was difficult to obtain a relevant and big enough dataset, the script used is a good solution to increase that number based on a single picture. To add to that, Tensorflow also provides some tools to distort and blur the pictures; thus, improving the possibility to increase the dataset. However, the representation of the two used components, paragraph and picture, is extremely simplistic and cannot be used to represent reality.

Secondly, regarding the component recognition, based on the dataset created, on the one hand, it was capable of recognizing the elements and give accurate probabilities as if an element was in the picture or not. On the other hand, the solution is not using Deep Learning; the state-of-the-art technology at the moment for the reason that I was unable to implement it. Thus, the algorithm is not able to learn by itself and requires to be trained again and again for each new category that we want to add.

Lastly, the code creator is in almost of the cases able to more or less understand that to generate HTML code, it needs to open and close tags and fill those tags with content. However, its result is completely random and even after the thorough discussion of the topic, a further control of its behavior during the generations hasn't been possible.

Based on those outcomes, I would say that my proof of concept is not complete but can be used as a boilerplate to explore new ways, new leads. For instance, the dataset creator can be improved further with blurs and distortions. To add to that, the component recognition should be implemented using Deep learning so that the model is able to learn by itself. And to finish, the code generator has to be researched more thoroughly in order to generate appropriate code depending on the elements found in the mock-up.

Another point that can be improved is the way each of the three parts communicate together because as of now, they work independently from each other.

More personally, I found working on this thesis extremely challenging as it was using state-of-the-art technologies and the topic was poorly documented leaving me with no guideline in researching and implementing a working solution. Also, the thesis process has been disturbing a lot because the more I was going forward, the more challenges I faced because of my lack of experience and knowledge. Still, the outcome of the thesis

remains, overall positive for me as it pushed me to investigate a topic that I would like to keep developing in the future.

References

AirBnb. (2018). *Sketching Interfaces Generating code from low fidelity wireframes*. Retrieved January 13, 2018, from Airbnb.design: https://airbnb.design/sketching-interfaces/

Anyoha, R. (2017, August 28). *The History of Artificial Intelligence*. Retrieved February 8, 2018, from Harvard: http://sitn.hms.harvard.edu/flash/2017/history-artificial-intelligence/

Beltramelli, T. (2017, September 19). *pix2code: Generating Code from a Graphical User Interface Screenshot*. Retrieved January 14, 2018, from arxiv: https://arxiv.org/pdf/1705.07962v2.pdf

Berkman, J. (2017, August 22). *Machine Learning vs. Deep Learning*. Retrieved February 11, 2018, from Datascience: https://www.datascience.com/blog/machine-learning-and-deep-learning-what-is-the-difference

Brownlee, J. (2016, August 16). *What is Deep Learning?* Retrieved February 17, 2018, from Machine Learning Mistery: https://machinelearningmastery.com/what-is-deep-learning/

Cheng, J. (2018). *thesis_POC*. Retrieved 2018, from https://github.com: https://github.com/chengjo0/thesis_POC

Dettmers, T. (2015, March 26). *Understanding Convolution in Deep Learning*. Retrieved March 25, 2018, from TimDettmers: http://timdettmers.com/2015/03/26/convolution-deep-learning/

Geitgey, A. (2014, May 5). *Machine Learning is Fun! The world's easiest introduction to Machine Learning*. Retrieved January 12, 2018, from Medium: https://medium.com/@ageitgey/machine-learning-is-fun-80ea3ec3c471

Getgey, A. (2016, January 3). *Machine Learning is Fun! Part 2 Using Machine Learning to generate Super Mario Maker levels*. Retrieved January 14, 2018, from Medium: https://medium.com/@ageitgey/machine-learning-is-fun-part-2-a26a10b68df3#.lbsa5her1

Getgey, A. (2016, June 13). *Machine Learning is Fun! Part 3: Deep Learning and Convolutional Neural Networks*. Retrieved April 4, 2018, from Medium: https://medium.com/@ageitgey/machine-learning-is-fun-part-3-deep-learning-and-convolutional-neural-networks-f40359318721

gk_. (2017, March 23). *freeCodeCamp*. Retrieved April 3, 2018, from Image Recognition Demystified: https://medium.freecodecamp.org/image-recognition-demystified-fc9c89b894ce

Johnson, N. (2018, February 1). *How the Fourth Industrial Revolution is Reinventing the Future of Jobs*. Retrieved February 2, 2018, from Salesforce:

https://www.salesforce.com/blog/2018/02/future-of-jobs-fourth-industrial-revolution.html

karpathy. (2016, April 30). *char-rnn*. Retrieved April 17, 2018, from https://github.com: https://github.com/karpathy/char-rnn

Keenan, T. (2018). *How Image Recognition Work*. Retrieved March 15, 2018, from upwork: https://www.upwork.com/hiring/data/how-image-recognition-works/

McClelland, C. (2017, December 4). *The Difference Between Artificial Intelligence, Machine Learning, and Deep Learning*. Retrieved February 13, 2018, from Medium: https://medium.com/iotforall/the-difference-between-artificial-intelligence-machine-learning-and-deep-learning-3aa67bff5991

Moujahid, A. (2016, June 26). *A Practical Introduction to Deep Learning with Caffe and Python*. Retrieved February 19, 2018, from adilmoujahid: http://adilmoujahid.com/posts/2016/06/introduction-deep-learning-python-caffe/

Ng, A. (2018). *Machine Learning*. Retrieved March 10, 2018, from Coursera: https://www.coursera.org/learn/machine-learning

Nielsen, M. (2017, December 2). *chap1.html*. Retrieved May 1, 2018, from neuralnetworksanddeeplearning.com: http://neuralnetworksanddeeplearning.com/chap1.html

Nielsen, M. (2017, December 2). *chap2.html*. Retrieved May 2, 2018, from neuralnetworksanddeeplearning.com: http://neuralnetworksanddeeplearning.com/chap2.html

Nielsen, M. (2017, December 2). *chap3.html*. Retrieved May 2, 2018, from neuralnetworksanddeeplearning.com: http://neuralnetworksanddeeplearning.com/chap3.html

Oxford Dictionnary. (2018). *Artificial Intelligence*. Retrieved January 20, 2018, from Oxford Dictionnary: https://en.oxforddictionaries.com/definition/artificial_intelligence

Pigeon, S. (2018). *Introduction to Gradient Descent*. Retrieved April 3, 2018, from Harder, Better, Faster, Stronger: https://hbfs.wordpress.com/2012/04/24/introduction-to-gradient-descent/

RichmondVale. (2016, August 12). *The Four Industrial Revolutions In a Glimpse*. Retrieved February 10, 2018, from Richmond Vale Academy: http://richmondvale.org/industrial-revolutions/

Scikit. (2018). *Scikit-Learn*. Retrieved January 20, 2018, from scikit-learn: http://scikit-learn.org/stable/

Seif, G. (2018, January 21). *Deep Learning For Image Recognition*. Retrieved March 27, 2018, from Towards Data Science: https://towardsdatascience.com/deep-learning-for-image-classification-why-its-challenging-where-we-ve-been-and-what-s-next-93b56948fcef

Techopedia. (2018). *Integrated Development Environment*. Retrieved April 11, 2018, from techopedia: https://www.techopedia.com/definition/26860/integrated-development-environment-ide

TensorFlow. (2018). *Get Started for Beginners*. Retrieved March 28, 2018, from Tensorflow: https://www.tensorflow.org/get_started/get_started_for_beginners

Tensorflow. (2018). *premade_estimators*. Retrieved March 20, 2018, from Tensorflow: https://www.tensorflow.org/get_started/premade_estimators

Torch. (2018). *torch*. Retrieved May 13, 2018, from torch.ch: http://torch.ch/

Wallner, E. (2018, January 9). *Turning Design Mockup Into Code With Deep Learning*. Retrieved January 13, 2018, from Floydhub: https://blog.floydhub.com/turning-design-mockups-into-code-with-deep-learning/

Vasconcellos, P. H. (2017, June 22). *Top 5 Python IDEs For Data Science*. Retrieved April 11, 2018, from DataCamp: https://www.datacamp.com/community/tutorials/data-science-python-ide

World Economic Forum. (2018). *center-for-the-fourth-industrial-revolution*. Retrieved April 5, 2018, from weforum: https://www.weforum.org/center-for-the-fourth-industrial-revolution/about/

Voskoglou, C. (2017, May 5). *What is the best programming language for Machine Learning?* Retrieved April 11, 2018, from Towards Data Science: https://towardsdatascience.com/what-is-the-best-programming-language-for-machine-learning-a745c156d6b7

Appendices

Appendix 1. Thesis Proof of Concept Link

To access the code written during the thesis, follow the link here-after:
https://github.com/chengjo0/thesis_POC