Expertise
and insight
for the future

Abdul Aziz Alzayed

AI-Based Welding Quality Detection

Metropolia University of Applied Sciences

Master of Engineering

Information Technology

Master's Thesis

9 September 2021

PREFACE

It was a hard challenge to Join metalworks vocational school after I have finished the university of software Eng. And that was because of the delaying 3 years of receiving it from Syria.

I felt disappointed, but after I received my bachelor's certificate after long waiting, the smile came back to my face, and I decided to use my new knowledge of metalworks to build this project.

For my father in the sky who spent his life helping me,

For my mother who did her best to make me warm and optimistic in every critical situation.

For my brothers and sisters who helped me in everything in this life, and they were beside me.

For my wife and my son who made the alienation easy for me.

For my managers in Peikko (Juha and Taru) who helped me so much and every person who gave me a hand in my life.

I want to thank you and present you to this success, which without you would have been so difficult.

Finland-Lahti, 29/6/2020

Abdul Aziz Alzayed

Author Title Number of Pages Date	Abdul Aziz Alzayed AI-Based Welding Quality Detection 60 pages 7 Sep 2021
Degree	Master of Engineering
Degree Program	Information Technology
Instructor(s)	Juha Airola: Project Manager. Ville Jääskeläinen: Principal Lecturer.

Artificial Intelligence (A.I.) is a multidisciplinary field whose goal is to automate activities that presently require human intelligence. Recent successes in A.I. include computerized medical diagnosticians and systems that automatically customize hardware to particular user requirements. The major problem areas addressed in A.I. can be summarized as Perception, Manipulation, Reasoning, Communication, and Learning. Perception is concerned with building models of the physical world from sensory input (visual, audio, etc.).

Many occupations nowadays need a professional person to check their products, and one of them is metal welding. After welding a professional person needed to check visually the quality of welding, if it has a good shape or no, and then the professional gives a report if the welding has been accepted or not.

With AI based solution one can make this procedure faster and the developed solution should improve itself each time one adds something new to its knowledge.

This study focuses on an AI image classification solution to check the welding quality, how it works to fulfil the function and the accuracy using the .Net environment (ASP.NET, MVC, ML.NET, Console App, .NET Framework), jQuery, JavaScript, Bootstrap.

The study went through many challenges in terms of collecting images, collecting references, and examining the results of images analysis, and the results were successful and correct in the cases that have been used.

Keywords	Welding, ML.NET, Image Classification, AI, .NET environment.

Contents

List of Figures

List of Codes

List of Abbreviations

AI	Artificial Intelligence
API	Application Programming Interface
DNN	Deep Neural Network
GPU	Graphics Processing Unit
ML	Machine Learning
ML Model	A machine learning model is a file that has been trained to recognize certain types of patterns. You train a model over a set of data, providing it an algorithm that it can use to reason over and learn from those data.
MVC	Models-Views-Controllers, ASP.NET protocol
NDT	non-destructive testing
RT	Radiographic Testing
UT	Ultrasonic Testing
VT	Visual inspection

1 Introduction

Artificial intelligence (AI) is a branch of computer science concerned with building smart machines or smart programs capable of performing tasks that typically require human intelligence. AI has become an essential thing in everyday work's life, and it has been used to do routine types of work tasks and solve a lot of problems within a short time that are hard to humans to do.

For Example, "a system, called the CheXpert, is an automated chest x-ray interpretation model that leverages artificial intelligence to review x-ray images. Developed by the Stanford Machine Learning Group, the model can determine what is and is not pneumonia on an x-ray**.** The study found that those ultra-quick findings may enable physicians reading X-rays to accurately confirm a pneumonia diagnosis significantly faster than current clinical practice, enabling treatment to start sooner, which is vital for severely ill patients who are suffering from pneumonia". [1]

Also, an example for everyday AI usage in the companies: sales predictions.
There are many studies on this matter, and one of the researchers referred briefly to its effectiveness and use, saying: "AI can be a force multiplier in terms of heightening the accuracy of sales forecasting. According to research from the Aberdeen Group, companies boasting accurate sales forecasts are 10% more likely to grow their revenue year-over-year and 7% more likely to hit quota. AI addresses many of the inherent flaws associated with the weighted pipeline and other traditional forecasting methods". [2]

CRM and X-ray diagnostic by AI are some examples of what can be solved in everyday life, so what if one would start to use it in other occupations such as metal welding?

Peikko Oy is a company for manufacturing building steel solutions [3], and one important thing in its production line is to take into account the quality of welding. The first step to verify the quality of welding is to make a visual check of the welding. Because Peikko is an international company, and the examination of the quality of welding in products in large numbers is cumbersome, especially because this work needs a professional in welding in order to check the quality. The matter is even more difficult in situations such

as the coronavirus crisis because it is often impossible for the professionals to go to the sites.

From these challenges come the research question of this thesis:
What if one would have a system that relies on artificial intelligence to classify the images and generate the report? Is it possible, and how it should be done? Can we start introducing AI technology in the factory?

AI image classification could solve the lack of professionals by giving the welder the needed result whether the welding is “Good” or “Bad”. By developing it further, it could give him/her the needed instruction to improve it, and it could accelerate and improve the process of quality. And the more it has samples in the dataset while building the model, the more precise results should come in the analysis.

This study explains that it is possible to create such a system (giving a report on whether welding is acceptable or not, i.e., “Good” or “Bad”.) using AI, how the AI model was built and consumed and how accurate it was based on this study. This study also explains how one can develop and expand the model further.

This thesis has been written in a simple and clear manner, without going into deep details to be understood by the employees of the company, and it has been divided into 7 main sections. The first section introduces the research problem and the company behind the study, the second gives a brief theoretical background of AI and its related topics, and the third and fourth sections show the building of the project and its results.

2 Method and Material

Because the project was new of its kind in the company, existing tools and software environments were used to ensure compatibility with the rest of the systems and programs. The environment was also selected according to past experience, ease of use, and consumption with other applications from the same environment.

The project used the "*ML.NET"* platform which is part of the "*.Net"* environment that the company uses as its software development platform. It can be a plugin in many "*.Net Core, .NET Framework*" projects which can be used with the desktop application or *ASP.NET* or *Blazor* website or any project that uses the same framework.

Like any AI image classification application, the project needs a set of samples for the cases that need to be detected and classified, and collecting the samples is the most important thing in the project. Usually, this kind of dataset can be found on websites such as "*Kaggle.com*" which is one of the websites that provides datasets of multiple cases. This website includes an extensive range of cases since it belongs to *Google*, but in our case, there was no dataset that had images that fulfilled our needs. For this reason, the company collected those images from many places, e.g. (company's factories, websites…etc.), and that is because the images must have a special shooting angle, so it was difficult to collect enough of them.

Used basic programming materials and libraries:

- *Asp.Net MVC C#* used for building the backend and the frontend of the website.

- *Azure Web App* used to host the website. And it was chosen because it contains a lot of features such as *Azure Insights App* for tracing the events and following up the errors, also the ability to insert slots to distribute the usage or publishing some testing version in these slots before swapping it to the production slot. The 64-bit type was used in its platform settings to allow the ML Model to be used.

- *AmChart.js* library, which is a JavaScript library for charts. The project used it to represent the results and make the report easy to read and understand.
- *C# console project* was created to build and generate the Model.

- *ML.NET* libraries were included to build the ML model.

- *Tensorflow.NET library* was used to implement the complete *Tensorflow API* in *C#* which allows *.NET* developers to develop, train and deploy Machine Learning models with the cross-platform *.NET* Standard framework.

- *JavaScript, CSS, jQuery* were used for UI design.

- *Ajax* was used to post and return values to UI.

- *Postman* was used to test the APIs.

The project passed into many stages, it started with building the ML Model, training the model, and evaluating it to have its accuracy, then showing how the Model can be consumed on the website to test the images and gives the results. And each stage is explained individually in the next Project Building section.

3 Current Welding Quality Testing

Testing the quality of welding has been developed many times since the welding invention has existed, and in recent years, the company has adopted several tests on welding to determine its quality. These tests are as follows:

- Vision Inspection
- Radiographic Testing (RT).
- Ultrasonic Testing.

These three types are used according to their order in the list, so the visual inspection is used first, then radiography, and finally an ultrasound test (sometimes ultrasound used before Xray and vice versa). They are the most common in factories worldwide and are approved for quality inspection in the production line.

These three tests discover welding problems extending from outside to inside, as shown in the following Figure 1.

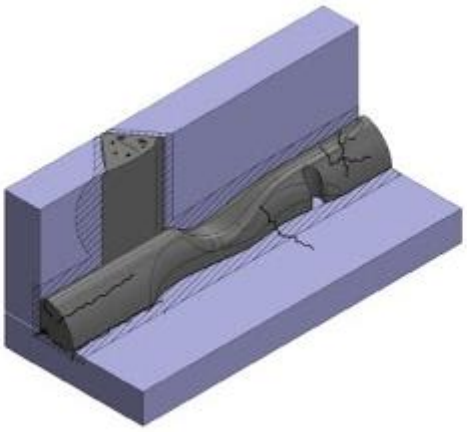

Figure 1 Oblique clip-on welding [4]

Figure1 shows the internal and external errors of welding, such as holes, cracks, warps, or cuts in the surface.

3.1 Vision Inspection (VT)

Visual inspection weld quality testing is a non-destructive testing (NDT) process where the weld is examined with eyes to determine discontinuity surfaces and welding dispersed dots etc. It is the most common method of weld quality testing and It is the first examination of the three tests previously mentioned, and it is used frequently.

This test requires a professional in welding to determine the welding assessment is acceptable or not.

An example of bad welding is shown in Figure 2. The welding is not continuous, and it has holes. This type of welding should not be accepted.

Figure 2 Bad Welding with holes inside

Here also in Figure 3 an example of good welding, the welding is continuous, with no holes and not a lot of dots. This type of welding should be accepted.

Figure 3 Good welding without holes

Visual Inspection becomes a hard task when a lot of products have been produced and the quality manager and the professionals, who may be busy with other tasks, need to check all of them. Also situations such as the Covid-19 virus, which restricts the movement of managers and the professionals to be present in the places of welding, is a serious challenge.

3.2 Radiographic Testing (RT)

This method of weld testing makes use of X-rays, produced by an X-ray tube, or gamma rays, produced by a radioactive isotope. The basic principle of radiographic inspection of welds is the same as that for medical radiography. Penetrating radiation is passed through a solid object, in this case a weld rather than part of the human body, onto a photographic film resulting in an image of the object's internal structure being deposited on the film.

The amount of energy absorbed by the object depends on its thickness and density. Energy not absorbed by the object will cause exposure of the radiographic film and these areas will be dark when the film is developed. Areas of the film exposed to less energy remain lighter.

Therefore, areas of the object where the thickness has been changed by discontinuities, such as porosity or cracks, will appear as dark outlines on the film. Inclusions of low density, such as slag, will appear as dark areas on the film while inclusions of high density, such as tungsten, will appear as light areas. All discontinuities are detected by viewing the shape and variation in density of the processed film.

Radiographic testing can provide a permanent film record of weld quality that is relatively easy to interpret by trained personnel. This testing method is usually suited to having access to both sides of the welded joint (except for double-wall signal image techniques used on some pipework). Although this is a slow and expensive method of nondestructive testing, it is an effective method for detecting porosity, inclusions, cracks, and voids in the interior of welds.

It is essential that qualified personnel conduct radiographic interpretation since false interpretation of radiographs can be expensive and interfere seriously with productivity. There are obvious safety considerations when conducting radiographic testing. X-ray and gamma radiation are invisible to the naked eye and can have serious health and safety implications. Only suitably trained and qualified personnel should practice this type of testing. [5]

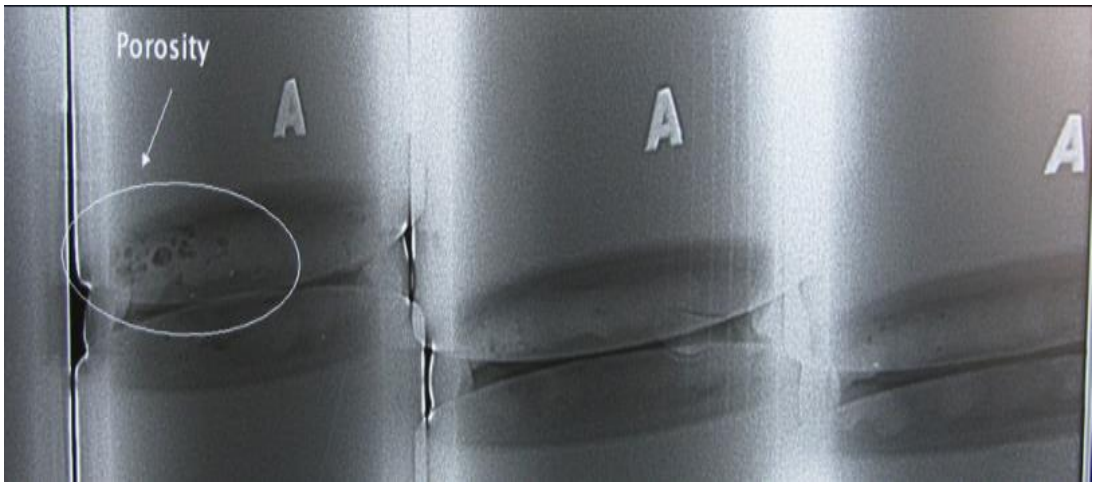

Figure 4 Radiographic Testing Example [6]

The above Figure 4 shows the presence of pores in the welding radiation image, which often causes a safety problem in the welding body.

The algorithms used in this thesis in detecting the visible welding images can also be used in analyzing the radiological images of welds in the future when one has collected sufficient radiographic images to train the ML Model.

3.3 Ultrasonic Testing (UT)

This method of testing makes use of mechanical vibrations similar to sound waves but of higher frequency. A beam of ultrasonic energy is directed into the object to be tested. This beam travels through the object with insignificant loss, except when it is intercepted and reflected by a discontinuity.

This system uses a transducer that changes electrical energy into mechanical energy. The transducer is excited by a high-frequency voltage, which causes a crystal to vibrate mechanically. The crystal probe becomes the source of ultrasonic mechanical vibration. These vibrations are transmitted into the test piece through a coupling fluid, usually, a film of oil called a Couplant.
When the pulse of ultrasonic waves strikes a discontinuity in the test piece, it is reflected back to its point of origin as shown in Figure 5. Thus, the energy returns to the transducer. The transducer now serves as a receiver for the reflected energy.

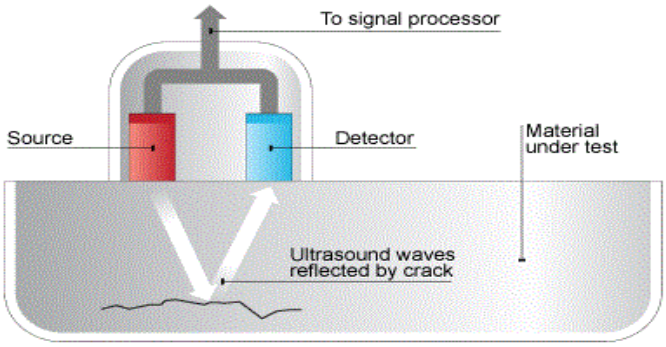

Figure 5 An illustrative example of Ultrasonic Testing mechanism [7]

The initial signal or main bang, the returned echoes from the discontinuities, and the echo of the rear surface of the test piece are all displayed by a trace on the screen of a cathode-ray oscilloscope, as shown in Figure 6.

Figure 6 An Ultrasonic device [8]

The detection, location, and evaluation of discontinuities become possible because the velocity of sound through a given material is nearly constant, making distance measurement possible, and the relative amplitude of a reflected pulse is more or less proportional to the size of the reflector as shown in Figure 7.

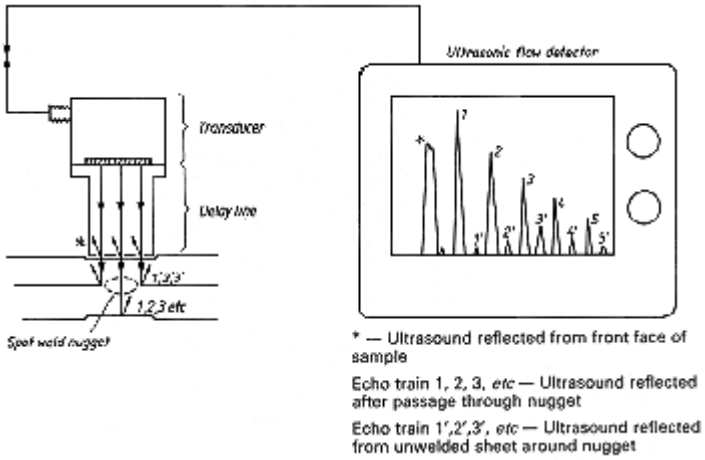

Figure 7 Reading the back waves [7]

One of the most useful characteristics of ultrasonic testing is its ability to determine the exact position of a discontinuity in a weld. This testing method requires a high level of operator training and competence and is dependent on the establishment and application of suitable testing procedures. This testing method can be used on ferrous and nonferrous materials, is often suited for testing thicker sections accessible from one side only, and can often detect finer lines or plainer defects which may not be as readily detected by radiographic testing. [4]

4 Theoretical Background

The focus of this section is to explain the used technologies in this project. First the basic concepts of Artificial Intelligence and Machine Learning are explained. Then the basic image classification methods are described. Finally, some commonly used tools and platforms are presented.

4.1 Artificial intelligent (AI)

Artificial Intelligence (**AI**) is a branch of computing that involves training computers to do things that normally require human intelligence. There is no formal agreed definition, and the definition differs from one scientific institute to another. And one of the reasons AI is so difficult to have a definition is because we still do not have a set definition or one solid concept for intelligence in general. Intelligence is often dependent on context.

It can be said, AI is one of the Computer sciences that uses Data Science and ML to build algorithms that can perform tasks in complex environments without constant guidance by a user and improving its performance by learning from experiences.

The following Euler diagram shows relationships between AI and other concepts related to Computer Science, Data Science, and other sciences under the umbrella of artificial intelligence which will be explained what used from them later:

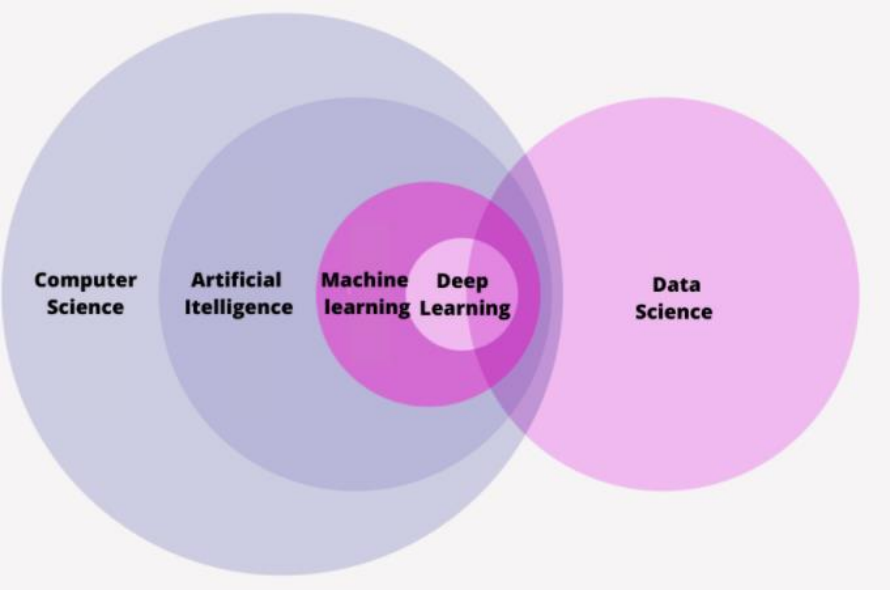

Figure 8 Euler diagram [9]

The keywords are shown in the Figure 8 diagram can briefly be defined as follows:

Machine learning (ML) can be said to be a subfield of AI, which itself is a subfield of computer science (such categories are often somewhat imprecise, and some parts of machine learning could be equally well or better belong to statistics). Machine learning enables AI solutions that are adaptive. A concise definition can be given as follows: Systems that improve their performance in a given task with more and more experience or data. [9]

Deep learning is an artificial intelligence (AI) function that imitates the workings of the human brain in processing data and creating patterns for use in decision making. Deep learning is a subset of machine learning in artificial intelligence that has networks capable of learning unsupervised from data that is unstructured or unlabeled. Also known as deep neural learning or deep neural network. It maps inputs to outputs and finds correlations, Also it is known as a "universal approximator" because it can learn to approximate an unknown function $f(x) = y$ between any input x and any output y, assuming they are related at all (by correlation or causation, for example). In the process of learning, a neural network finds the right *f*, or the correct manner of transforming *x* into *y*, whether that be $f(x) = 3x + 12$ *or* $f(x) = 9x - 0.1$.

Also the expression of "*Deep learning"* used to refer to "*stacked neural networks*" that is networks composed of several layers.

The layers are made of nodes, and the node is just a place where computation happens, loosely patterned on a neuron in the human brain, which fires when it encounters sufficient stimuli. A node combines input from the data with a set of coefficients, or weights, that either amplify or dampen that input, thereby assigning significance to inputs with regard to the task the algorithm is trying to learn, e.g. which input is most helpful is classifying data without error? These input-weight products are summed and then the sum is passed through a node's so-called activation function, to determine whether and to what extent that signal should progress further through the network to affect the ultimate outcome, say, an act of classification. If the signals pass through, the neuron has been "activated."

Here is a diagram of what one node might look like [10]:

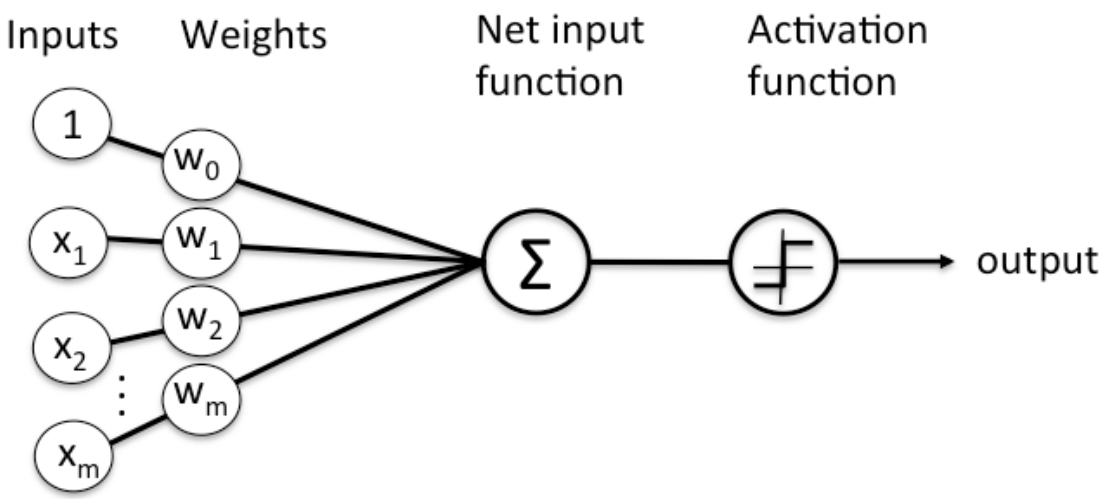

Figure 9 Neural Network Elements [10]

Data science is a recent umbrella term (a term that covers several subdisciplines) that includes machine learning and statistics, certain aspects of computer science including algorithms, data storage, and web application development. Data science is also a practical discipline that requires an understanding of the domain in which it is applied, for example, business or science: its purpose, basic assumptions, and constraints. Data science solutions often involve at least a pinch of AI (but usually not as much as one would expect from the headlines). [9]

And since this thesis uses Machine Learning to process images as data, and use both to solve a problem, this thesis is one example of Artificial Intelligence usage. also can be said about the difference between the two fields to make the idea easy and obvious, as follows:

4.2 Machine Learning (ML)

Machine learning has been defined before, but to make its definition clearer, it could be defined as follows:
“Is a method of data analysis that automates analytical model building. It is a branch of artificial intelligence based on the idea that systems can learn from data, identify patterns and make decisions with minimal human intervention.”

The roots of machine learning are in statistics, which can also be thought of as the art of extracting knowledge from data. Especially methods such as **linear regression** and **Bayesian statistics**, which are both already more than two centuries old (!), are even today at the heart of machine learning. [5]

The area of Machine Learning is often divided into subareas according to the kinds of problems being solved. A rough categorization is as follows:

1- Supervised learning.
2- Unsupervised learning.
3- Reinforcement learning.

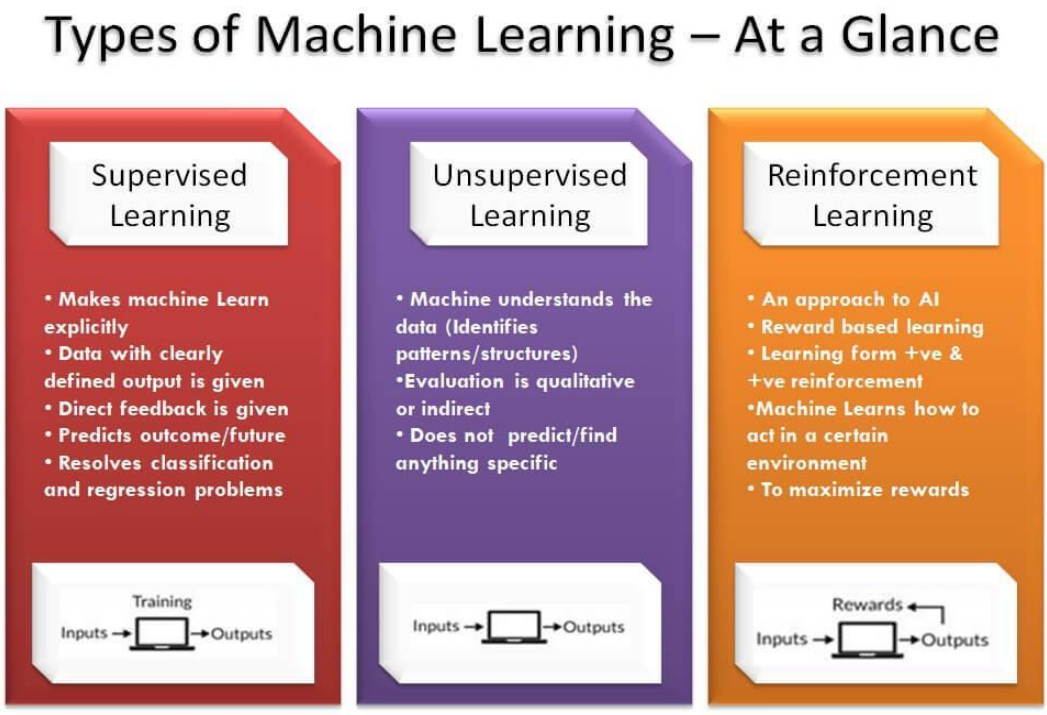

Figure 10 Types of ML at a glance [11]

Each type will be explained briefly to distinguish each type from others, but the focus will be on the first type (Supervised learning) as it is used in this thesis. and the reasons for choosing this type will be mentioned in the **ML-Supervised learning** section.

4.2.1 Type 1 - Supervised learning:

The Inputs have been given for example photographs with cats and dogs and the task is to predict the correct output or label, for example, which animal is in the picture (cats or dogs.). In the simplest cases, the answers are in the form of yes/no (we call these binary classification problems).

Most of the practical machine learning uses supervised learning.
Supervised learning is where having input variables (x) and an output variable (Y) and an algorithm has been used to learn the mapping function from the input to the output.

$$Y = f(X)$$

The goal is to approximate the mapping function so well that when having new input data (x) that can predict the output variables (Y) for that data.

It is called supervised learning because the process of an algorithm learning from the training dataset can be thought of as a teacher supervising the learning process. We know the correct answers, the algorithm iteratively makes predictions on the training data, and is corrected by the teacher. Learning stops when the algorithm achieves an acceptable level of performance.

Supervised learning problems can be further grouped into regression and classification problems:

- **Classification:** classification used when the output variable is a category, such as “Good Welding” or “Bad Welding” or “doted Welding” and “catted Welding”.
- **Regression:** regression used when the output variable is a real value, such as “dollars” or “weight”. [12]

And this thesis has used this concept to solve its project by giving the image as (X) variable and have the results as (Y) variable, and the equation of Y=f(X) can be done when the function is the *Classification algorithm*.

4.2.2 Type 2- Unsupervised learning:

There are no labels or correct outputs, where the labels are the name of the categories that the data belong to, e.g. dog, cat. The task is to discover the structure of the data: for example, grouping similar items to form “clusters”, or reducing the data to a small number of important “dimensions”. Data visualization can also be considered unsupervised learning.

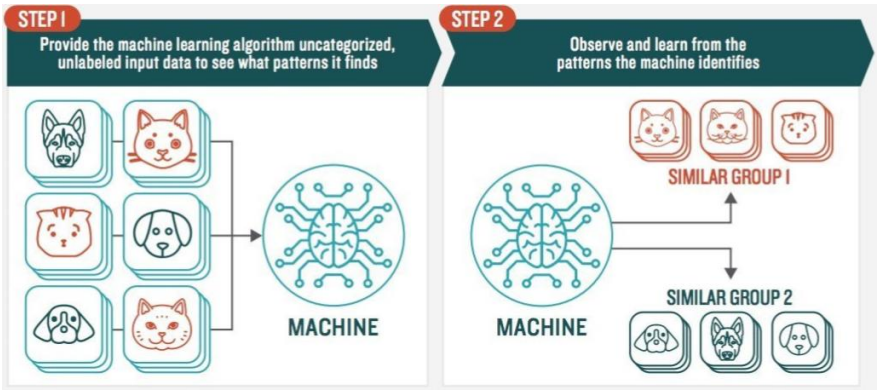

Figure 11 Unsupervised learning [13]

4.2.3 Type 3 - Reinforcement learning:

Reinforcement Learning is another part of Machine Learning that is gaining a lot of prestige in how it helps the machine learn from its progress. Readers who have studied psychology in college would be able to relate to this concept on a better level. Reinforcement Learning spurs off from the concept of Unsupervised Learning and gives a high sphere of control to software agents and machines to determine what the ideal behavior within a context can be. This link is formed to maximize the performance of the machine in a way that helps it to grow. Simple feedback that informs the machine about its progress is required here to help the machine learn its behavior.

Reinforcement Learning is not simple and is tackled by a plethora of different algorithms. As a matter of fact, in Reinforcement Learning an agent decides the best action based on the current state of the results.

The growth in Reinforcement Learning has led to the production of a wide variety of algorithms that help machines learn the outcome of what they are doing. Since we have a basic understanding of Reinforcement Learning by now, we can get a better grasp by forming a comparative analysis between Reinforcement Learning and the concepts of Supervised and Unsupervised Learning that we have studied in detail before. [14]

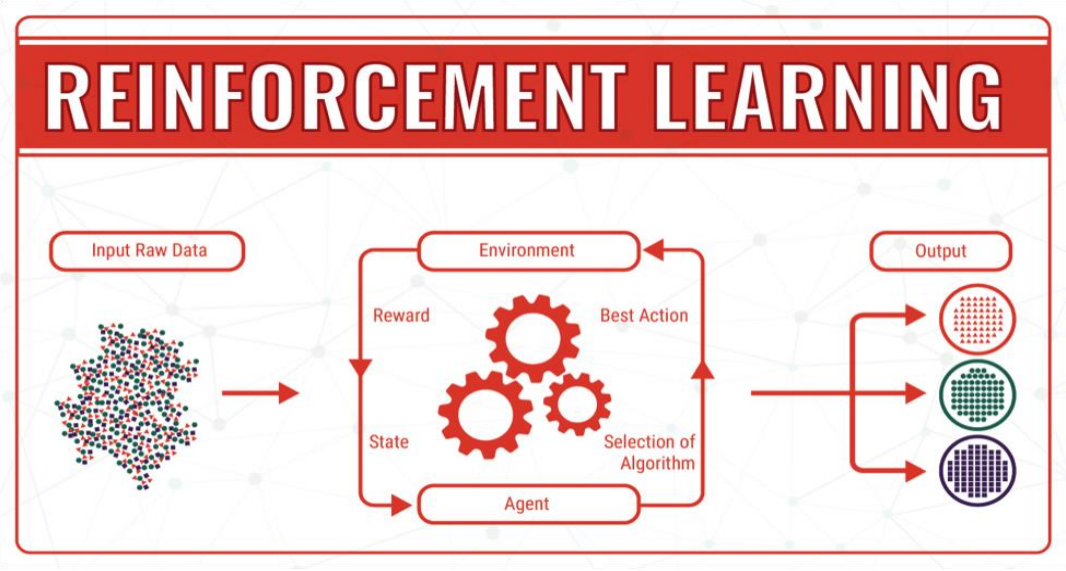

Figure 12 Reinforcement learning [14]

4.3 Image Classification in ML

Classification is a supervised machine learning approach, in which the algorithm learns from the data input provided to it, and then uses this learning to *classify new observations*. In other words, the training dataset is employed to obtain better boundary conditions which can be used to determine each target class; once such boundary conditions are determined, the next task is to predict the target class.

As has been mentioned before, Supervised machine learning categorizes into *Regression* and *Classification*. The *Regression* technique can be used to predict the target values of continuous variables, like predicting the salary of an employee. In contrast, the *Classification* technique has been used for predicting the class labels for given input data.

And the *Classification* technique has the types:

- *Binary* classifiers work with only two classes or possible outcomes (example: positive or negative sentiment; whether the lender will pay a loan or not; etc.),

- and *Multiclass* classifiers work with multiple classes (ex: to which country a flag belongs, whether an image is an apple or banana or orange; etc.). Multiclass assumes that each sample is assigned to one and only one label.

Binary Classification is good for simple cases, Example: Check email is spam or not, predicting gender based on height and weight which have only two classes, and the *Multi-Class Classification* algorithm implemented performs well on problems with a large number of features.

And since the working is with 2 classes “good and bad”, the algorithm supposed to be Binary Classification, but ML.Net image classification’s scenario only provides the multi-class classification, so it has been used since it full fill the task. The algorithm is

also good later when the categories of the images are increased to more than 2 (good-bad).

Also *ML.NET* used the “one vs all” technique as a classify model. which it’s a heuristic method for using binary classification algorithms for multi-class classification.

It involves splitting the multi-class dataset into multiple binary classification problems. A binary classifier is then trained on each binary classification problem and predictions are made using the model that is the most confident.
For example, given a multi-class classification problem with examples for each class ‘red,’ ‘blue,’ and ‘green‘. This could be divided into three binary classification datasets as follows:

- Binary Classification Problem 1: red vs [blue, green]
- Binary Classification Problem 2: blue vs [red, green]
- Binary Classification Problem 3: green vs [red, blue] [15] [16]

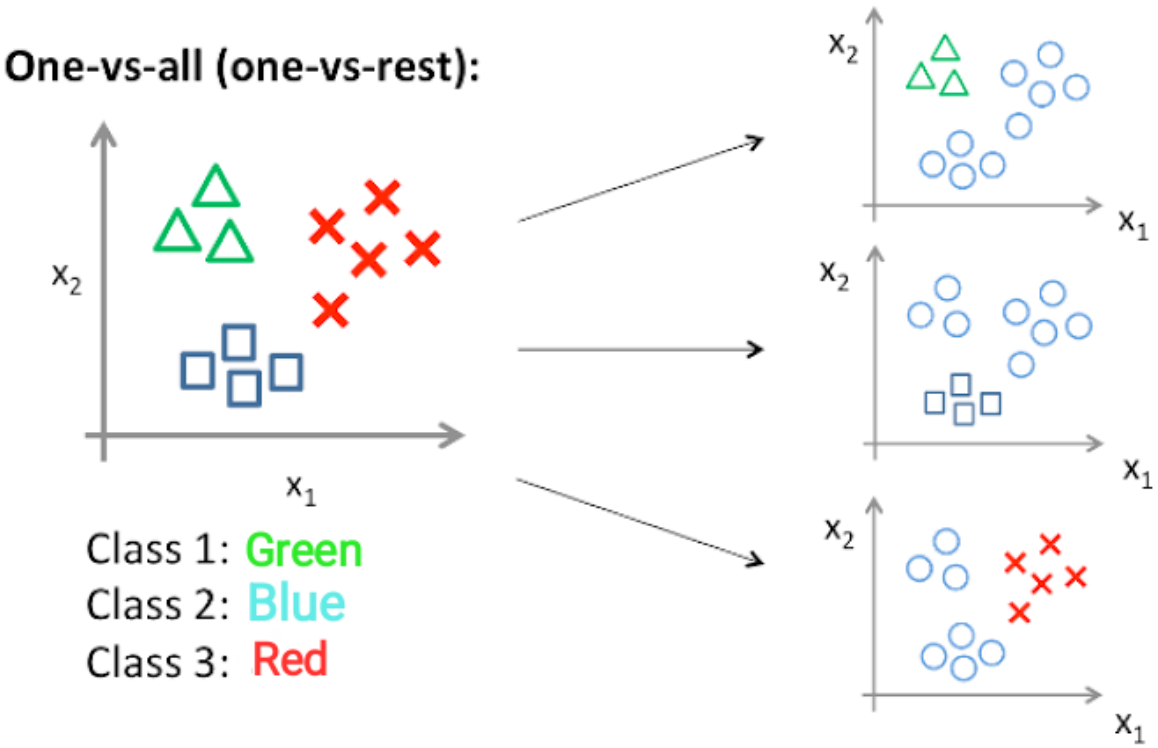

Figure 13 One-vs-all model [17]

The following figure illustrates images classifier scenario, where multiple different images are entered into a pre-trained Deep learning model and this model classifies the images into different predefined classes.

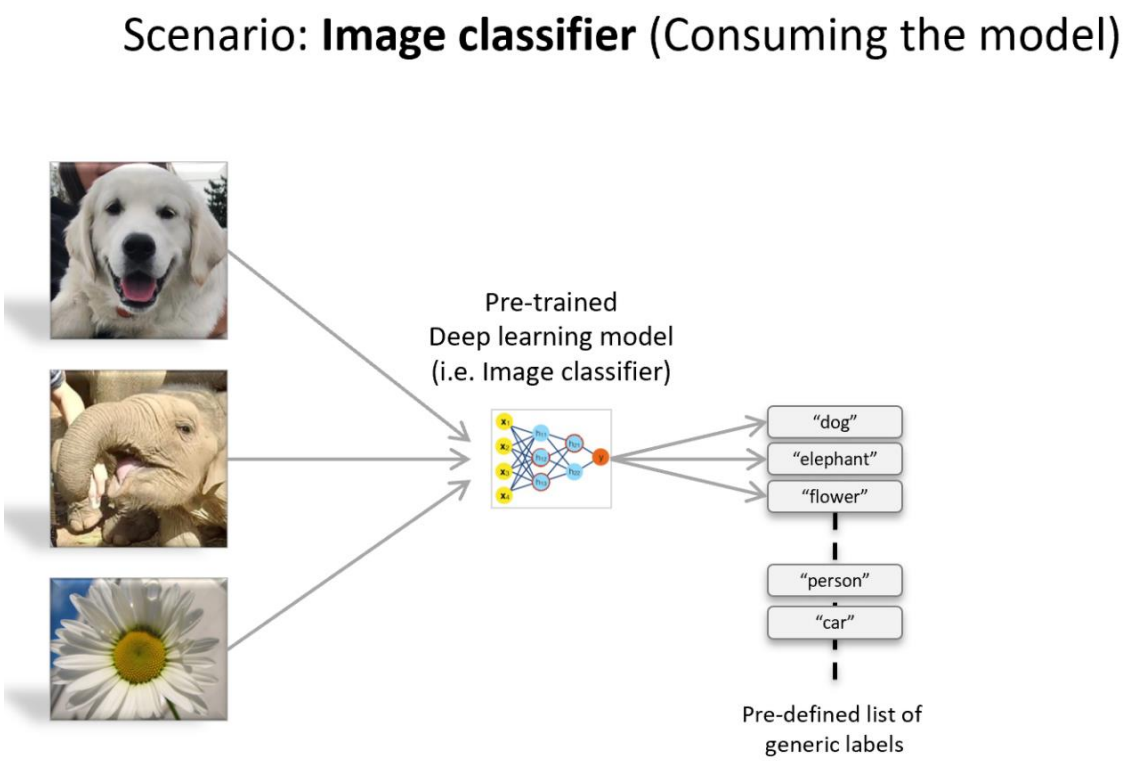

Figure 14 pre-trained Deep Learning model [18]

For more details about deep learning in *Tensorflow*, an article has been written by "*Richmond Alke*" in the "*Towards data science*" corporation website, that gives an overview of Deep Learning and the algorithms that have been used in *Tensorflow* libraries (which have been used in this project). [19]

4.4 ML Model Pipeline

A machine learning pipeline is a way to codify and automate the workflow it takes to produce a machine learning model. Machine learning pipelines consist of multiple sequential steps that do everything from data extraction and pre-processing to model training and deployment. [20]

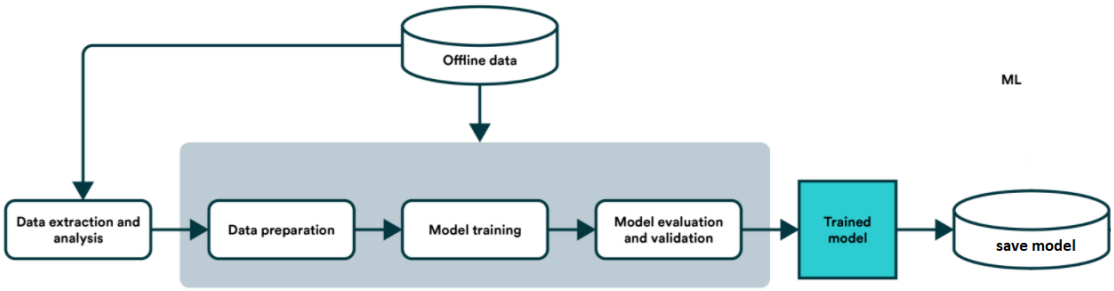

Figure 15 ML Pipeline [21]

The gray area in Figure 15 shows the structure of the pipeline in general which will be explained later in creating the pipeline section.

4.5 Transfer Learning

The definition of ‘transfer learning’ is the following:
“Transfer learning is a machine learning method where a model developed for an original task is reused as the starting point for a model on a second different but related task. For example, the knowledge gained while learning to recognize cars could apply when trying to recognize trucks.”

Transfer learning applies knowledge gained from solving one problem to another related problem.

Training a deep learning model from scratch requires setting several parameters, a large amount of labeled training data, and a vast amount of computing resources (hundreds of **GPU** hours) as shown in Figure 16. Using a pre-trained model along with transfer learning allows shortcutting the training process.

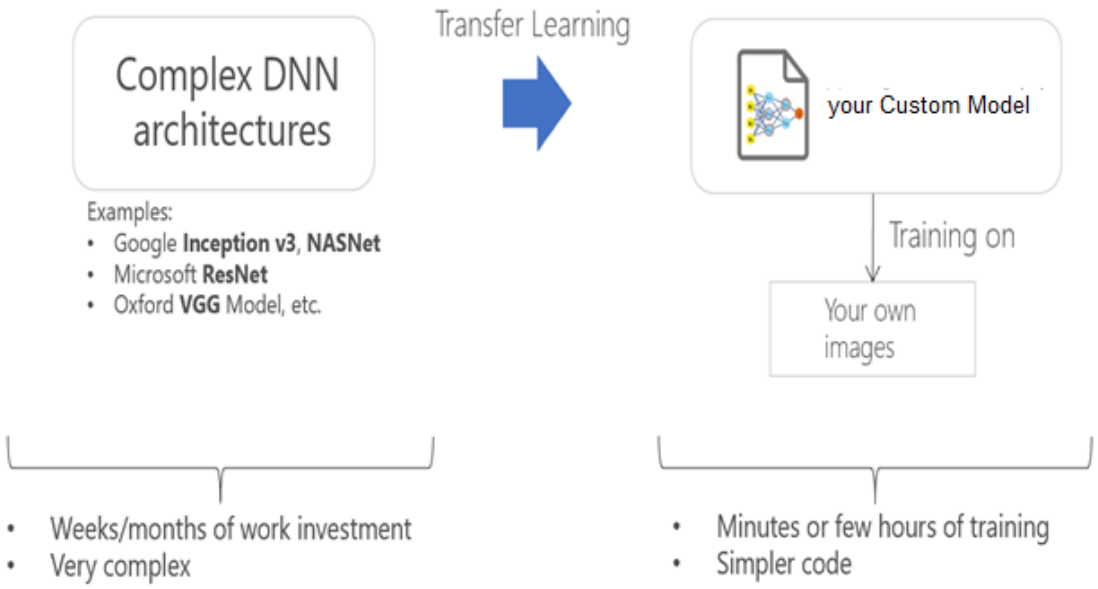

Figure 16 Transfer Learning [20]

4.5.1 Benefits of Native DNN Transfer Learning in ML.NET

The main benefit provided by the 'Transfer Learning' approach is *Full optimization power within the Deep Neural Network(DNN)framework*. Transfer Learning happens within *TensorFlow DNN* models, the ML.NET team will be able to optimize the retraining process with many improvements such as re-train one or more layers within the DNN graph plus any other tuning within the TensorFlow graph.

Here are simplified diagrams on how transfer learning happens under the covers when using the ML.NET *ImageClassification* estimator. Those graph diagrams are simplified diagrams taken from screenshots while using the *Netron* tool after opening the serialized TensorFlow .pb models: [18]

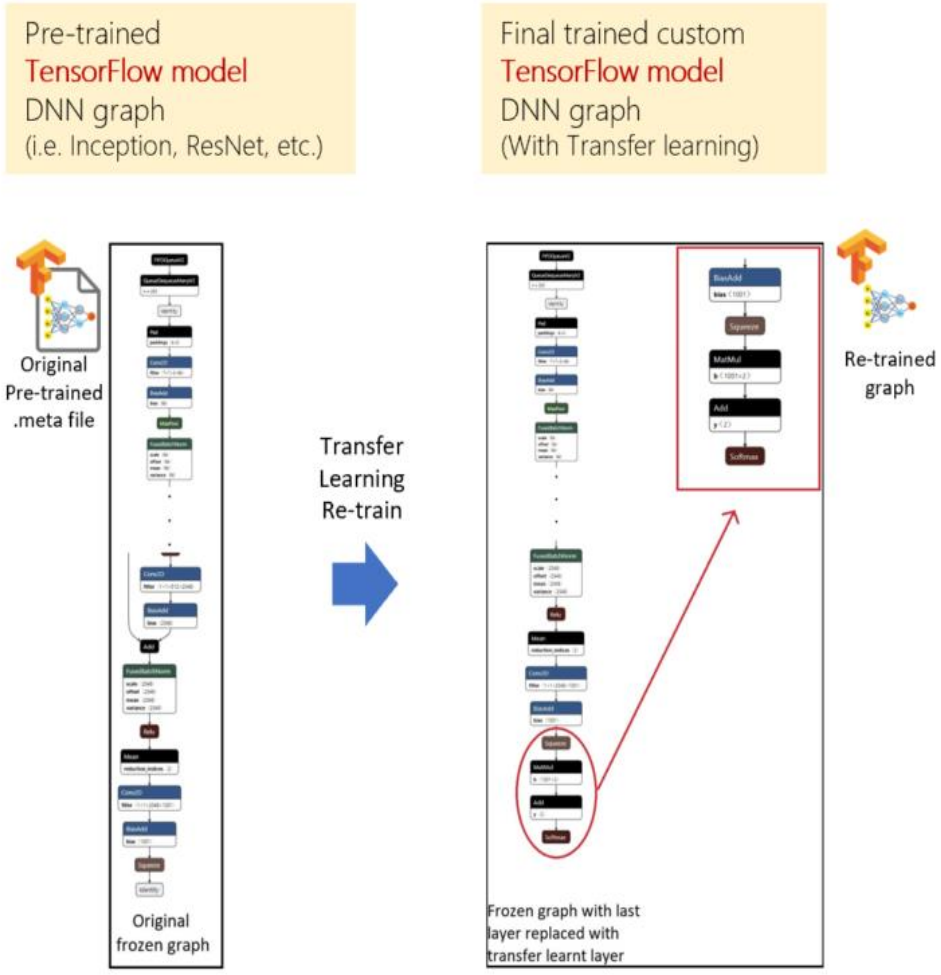

Figure 17 DNN Transfer Learning [18]

4.5.2 Simple API encapsulating DNN transfer learning

The first main benefit of this new *ImageClassification* API in ML.NET is simplicity.

It is not just a scenario-oriented API for image classification/recognition. Microsoft pushing the limits and they basically shrank hundreds of lines of code using the TensorFlow.NET bindings for C# and surface a very simple and easy to use API for Image Classification meaning that in a couple of lines one can implement a model training which is internally doing a native TensorFlow training, as illustrated in the following diagram:

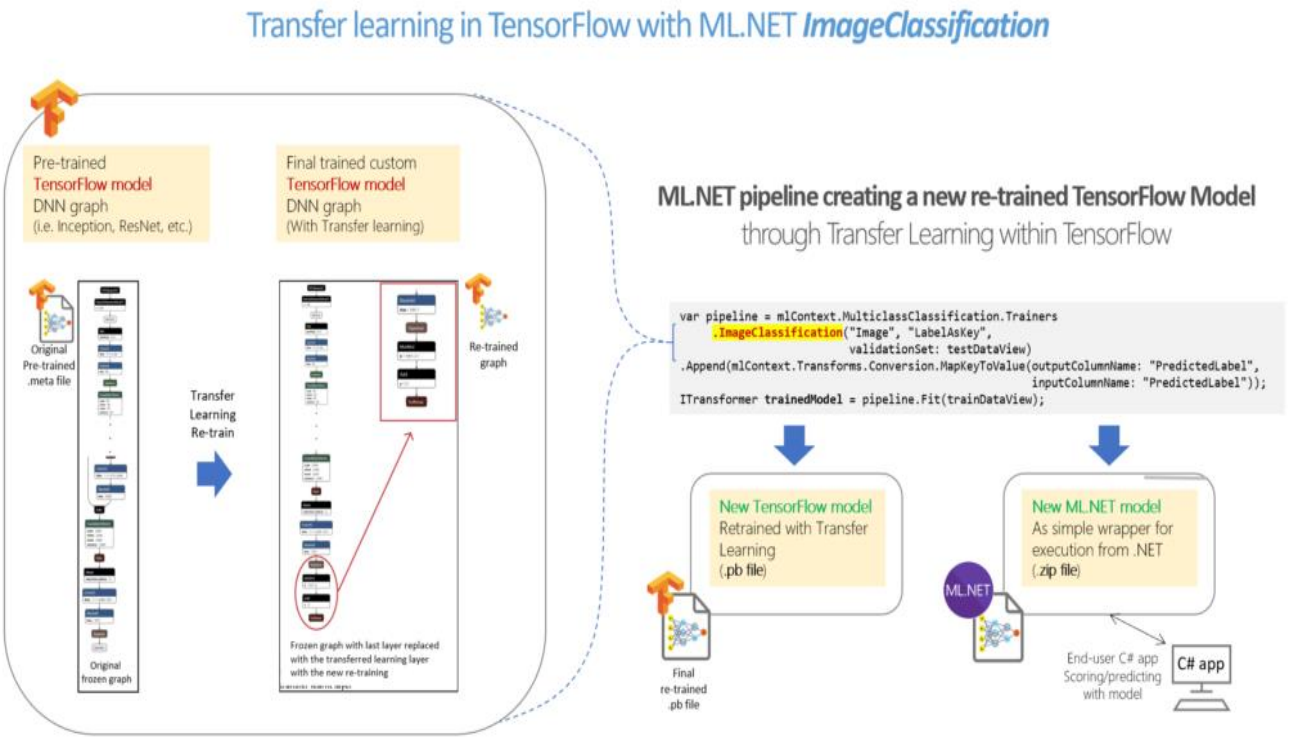

Figure 18 TensorFlow With ML.NET ImageClassification [18]

4.5.3 TensorFlow pre-trained model (DNN architecture)

A pre-trained Deep Learning model (DNN architecture) is simply used to generate features from all images with traditional ML.NET algorithms (using a multi-class classification ML Task trainer such as the ImageClassificationTrainer).

In more detail, it has used the Inception model as a featurizer. This means that the model will process input images through the neural network, and then it will use the output of the tensor which precedes the classification. This tensor contains the image features, which allows to identify an image.

Finally, these image features will feed into an ImageClassificationTrainer algorithm/trainer which will learn how to classify different sets of image features.

That approach in a visual illustration below:

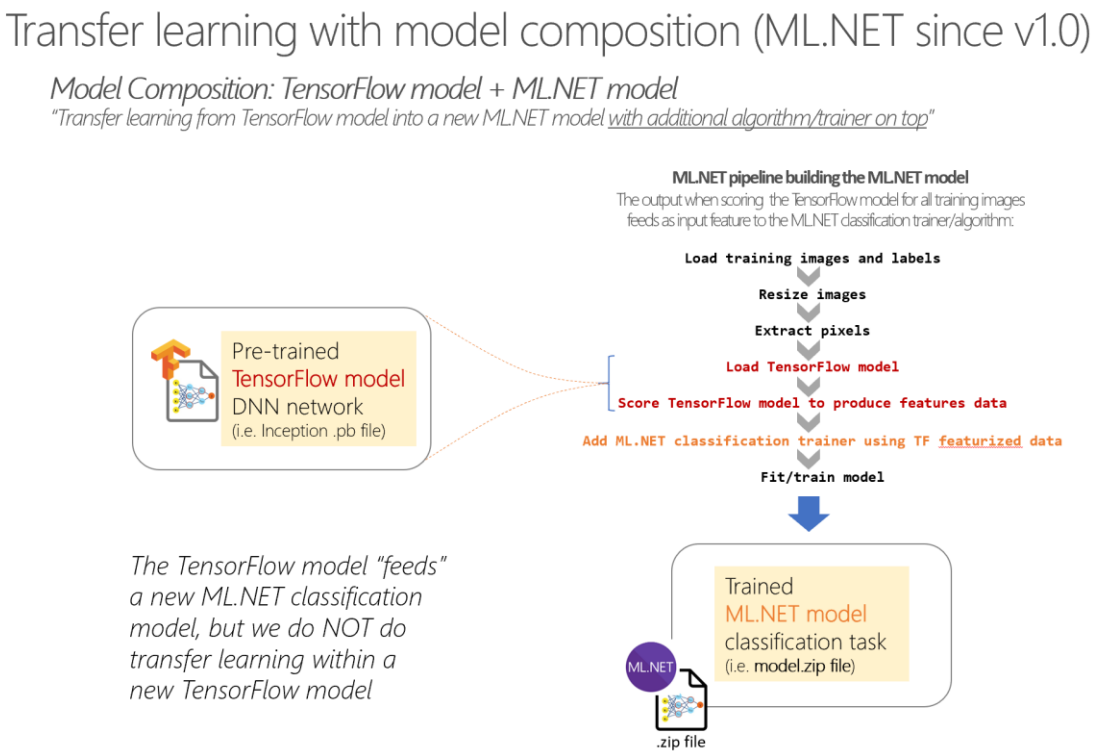

Figure 19 TensorFlow With ML.NET Transfer Learning [18]

In this case, the training not in TensorFlow but simply using a TensorFlow pre-trained model as featurizer to feed a regular ML.NET algorithm and therefore the only thing that is produced is a ML.NET model but not a new retrained TensorFlow model. [18]

There is a simple and realistic article that explains the used mathematical models and the algorithms in processing the images and classify them that used in TensorFlow. [22]

After the Theoretical section has explained in brief the used technology in theory in order to give an overview of what is this technology consists of, the next Project Building section will show how the project has been built to perform its task.

5 Project Building

This section will explain the building stages of this project, how the ML Model has been built and saved and used to evaluate the images, to give an overview of how ML can be initialized and used in the *.NET* environment.

The structure of the project divided into two main sections *Build Model* and *Use Model* as follows:

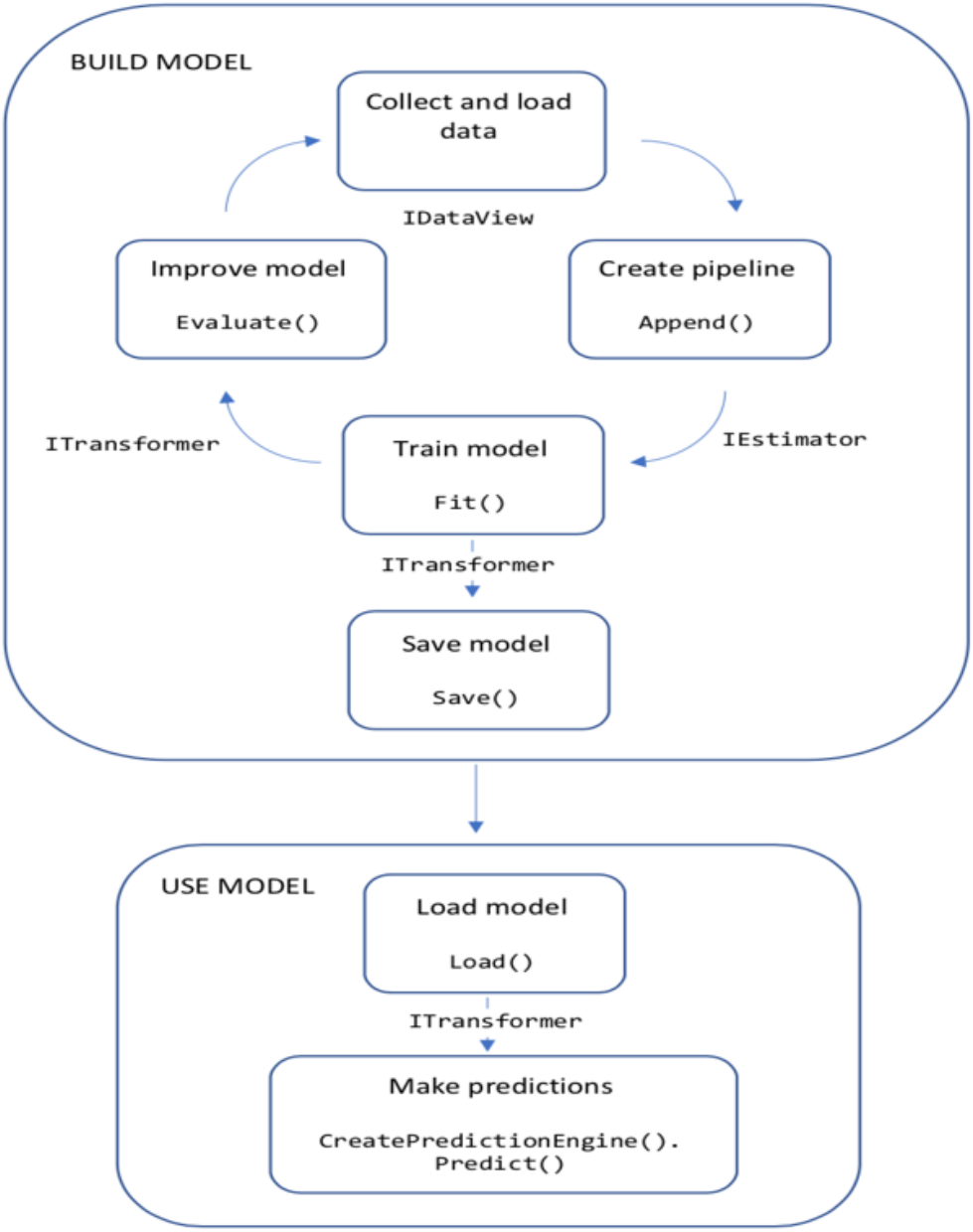

Figure 20 Project Building [23]

5.1 Model Building

ML.NET has been used to build the *ML Model* of this project, and ML.NET is a machine learning model for *.NET* developers. It can be used for many scenarios, such as sentiment analysis, price prediction, product recommendation, sales forecasting, image classification, object detection.
It can be trained as a custom model by specifying an algorithm or can import pre-trained TensorFlow and ONNX models.

The ML Model is the brain, and it is an important part of the project, this section will be explained in detail how to build it by using the algorithms that have been mentioned above. And how to train, evaluate, and use it.

Model building passes in many levels:

1. Creating ML.NET Context.
2. Collect and load data.
3. Creating pipeline.
4. Train Model.
5. Improve Model (Evaluate).
6. Save Model.

5.1.1 Creating ML.NET Context

MLContext is the starting point for all *ML.NET* operations. It's used for all aspects of creating and consuming ML.NET models. It is similar conceptually to *DbContext* in Entity Framework.

```
var mlContext = new MLContext();
```

Also, it's the common context for all *ML.NET* operations. Once instantiated by the user, it provides a way to create components for data preparation, feature engineering, training, prediction, model evaluation. It also allows logging, execution control, and the ability to set repeatable random numbers as shown in Figure 21.
Once trained, it will be possible to test the model for accuracy, saving it to a disk or a place in the cloud, and using it to make predictions. *MLContext* can be initialized from a model that was previously saved to a disk or a cloud.

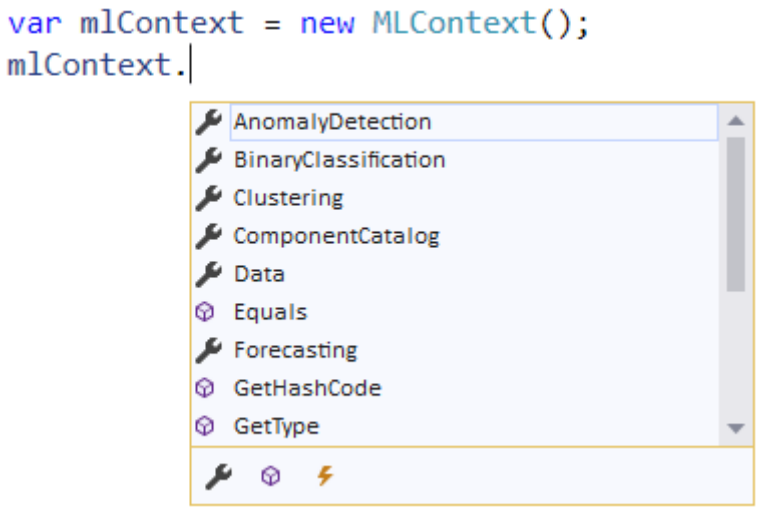

Figure 21 MLContext

Once the Libraries of ML.NET have been referenced, the first step is to prepare and collect the data and feed the context with it, and this is what the next section explains.

5.1.2 Collecting and loading data

The images were in color and have been collected from different locations with different resolutions and they were about 600 images, each class 300, and the manipulation was cutting the unnecessary edges, and reduce the resolution if it's big.

As long as the Model is built as a supervised ML, it has learned the behaviors that the project needs to use, by fed it with suit Input data to its task. And the Input data has been prepared in the following steps:

1- The Input data has been fed with Features (images) and their labels (good and bad).
2- ML uses labels to classify the images that have been fed and by both the labels and images the model will be trained which will be explained in the Training Model section.
3- The data has been distributed in different named folders (Good Welding Folder and Bad Welding Folder), the bad weld images have been saved in the Bad folder and the good well images have been saved in the Good folder.
4- The images have been used as Features and the *names of the folders* have been used as Labels for the Model.
5- Then running a method to save the labels and the paths of the images in a *.tsv* file to be loaded and trained later. (Figure 22 - 23)

```
Label   ImageSource
BAD C:\assets\BAD\Bad (1).jpg
BAD C:\assets\BAD\Bad (10).jpg
BAD C:\assets\BAD\Bad (100).jpg
BAD C:\assets\BAD\Bad (101).jpg
BAD C:\assets\BAD\Bad (102).jpg
BAD C:\assets\BAD\Bad (103).jpg
BAD C:\assets\BAD\Bad (104).jpg
BAD C:\assets\BAD\Bad (105).jpg
BAD C:\assets\BAD\Bad (106).jpg
BAD C:\assets\BAD\Bad (107).jpg
BAD C:\assets\BAD\Bad (108).jpg
BAD C:\assets\BAD\Bad (109).jpg
BAD C:\assets\BAD\Bad (11).jpg
```

Figure 22 TSV file

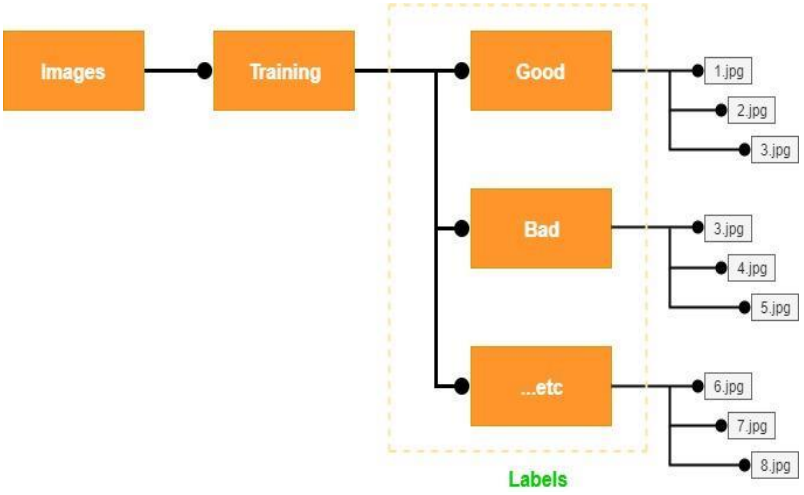

Figure 23 Saving images example in the folders

After the Input data has been collected in the TSV file, *ML Context* has loaded the data into *IDataView*. Where *IDataView* represented the Input Data in *ML.NEt*, it is a flexible, efficient way of describing tabular data (for example, rows and columns). *IDataView* objects can contain numbers, text, Booleans, vectors, and more.

```
IDataView trainingDataView = mlContext.Data.LoadFromTextFile<ModelInput>(
                                        path: TRAIN_DATA_FILEPATH,
                                        hasHeader: true,
                                        separatorChar: '\t',
                                        allowQuoting: true,
                                        allowSparse: false);
```

Code 1 Loading images paths from a CSV file

Collecting images as samples is not an easy task, the images must be carefully selected away from interference, so when taking the picture, it should be as much as possible only for welding and around it and not for the entire body of the piece, and it must be focused neither far nor close. In other words, the samples must be clear and free of noise, also taking them in all welding positions vertical, horizontal... etc.

The more samples collected, the better the model is trained, and thus it should have better accuracy when it's used.

These what we called avoiding overfitting and underfitting in ML training. [24]
Figure 24 shows an example of a clear image of welding that shows the welding clearly away from other objects with good resolution:

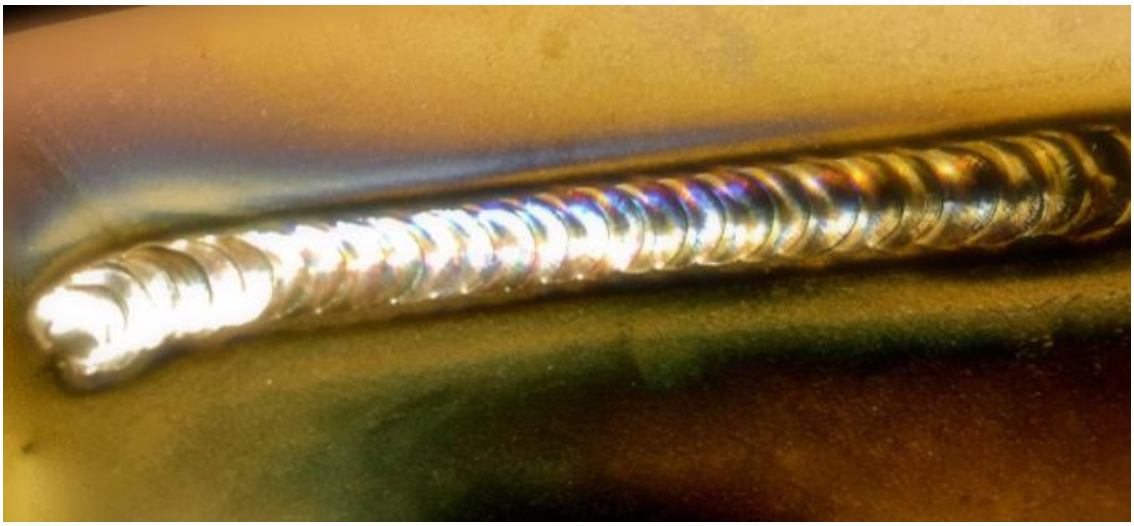

Figure 24 Clear Image Example

5.1.3 Creating pipeline

Once the samples (data) have been collected and loaded into *MLContext*, building the pipeline has been started to extract features and apply a machine learning algorithm. And Figure 25 shows the main steps that the creation of pipeline has been followed:

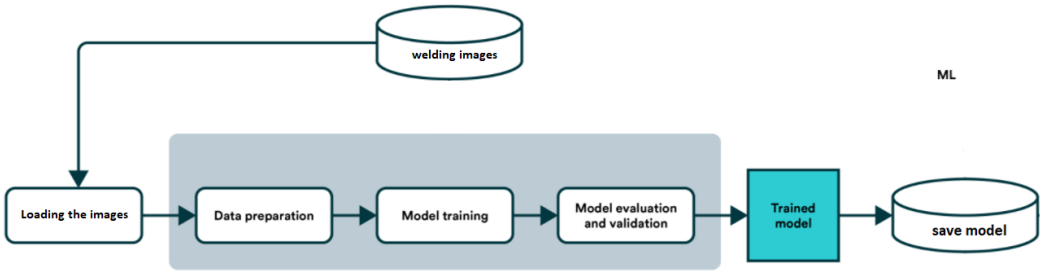

Figure 25 Created pipeline

```
IEstimator<ITransformer> trainingPipeline = BuildTrainingPipeline(mlContext);

public static IEstimator<ITransformer> BuildTrainingPipeline(MLContext mlContext)
        {
            try
            {
                // Data process configuration with pipeline data transformations
                // Apply transforms to the input dataset:
                // MapValueToKey : map 'string' type labels to keys
                // LoadImages : load raw images to "Image" column
                            var dataProcessPipeline = mlContext.Transforms.Conver-
                sion.MapValueToKey("Label", "Label")
                                        .Append(mlContext.Transforms.LoadRawImage-
                Bytes("ImageSource_featurized", null, "ImageSource"))
                                                .Append(mlContext.Trans-
                forms.CopyColumns("Features", "ImageSource_featurized"));
                   // Set the training algorithm
                var trainer = mlContext.MulticlassClassification.Trainers.ImageClassifi-
                cation(new ImageClassificationTrainer.Options() { LabelColumnName = "La-
                bel", FeatureColumnName = "Features" })
                                                .Append(mlContext.Trans-
                forms.Conversion.MapKeyToValue("PredictedLabel", "PredictedLabel"));

                   var trainingPipeline = dataProcessPipeline.Append(trainer);

                   return trainingPipeline;
                }
                catch (Exception)
                {

                   throw;
                }
        }
```

Code 2 C# script for building and training the pipeline

Code 1 shows C# coding follow of how the pipeline has been built and trained, and the meaning of keywords are as follow:

- **Map Value To Key** converts categorical values into keys. e.g.” `bad, good`”.
- **Load Raw Image Bytes** loads the data from the column ”`ImageSource`” as an image of raw bytes to a new column ”`ImageSource_featurized`”.
- **Copy Columns** copies the data from the column “`ImageSource_featurized`” to a new column ”`Features`”.
- **Multiclass-classification** algorithm has been chosen as what the code shows.
- **Image classification** is a computer vision problem. It takes an image as input and categorizes it into a prescribed class, and it is probably the most common task in *Deep Learning.*

Transfer Learning has been used here by calling *Image Classification API*. It uses *TensorFlow.NET,* a low-level library that provides C# bindings for the *TensorFlow C++ API.*

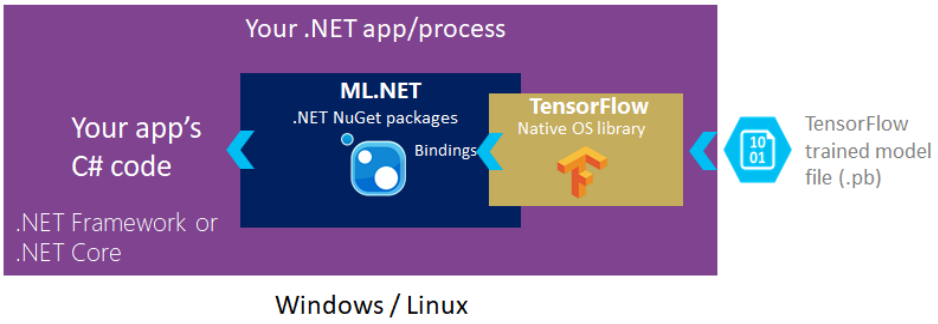

Figure 26 TensorFlow Libraries [18]

The above figure shows how simply a reference can be added to the *ML.NET NuGet packages* in the *.Net core* or *.Net framework* apps. Under the covers, *ML.NET* includes and references the native TensorFlow library which allows the coder to write code that loads an existing trained TensorFlow model file for scoring.

5.1.4 Train Model

The data transformations and algorithms that have been specified (Multi-class Classification algorithm) are not executed until calling the *Fit* method (because of *ML.NET*'s lazy loading approach).

```
// Train Model
ITransformer mlModel = TrainModel(mlContext, trainingDataView, trainingPipeline);

public static ITransformer TrainModel(MLContext mlContext, IDataView trainingDataView,
IEstimator<ITransformer> trainingPipeline)
{
    try
    {
      Console.WriteLine("=============== Training  model ===============");

      ITransformer model = trainingPipeline.Fit(trainingDataView);

      Console.WriteLine("=============== End of training process ===============");
                return model;
     }
            catch (Exception)
     {

                throw;

     }
}
```

Code 3 Train Model method

The Image Classification API starts the training process by loading a pre-trained TensorFlow model. The training process consists of two steps:

1- Bottleneck phase
2- Training phase

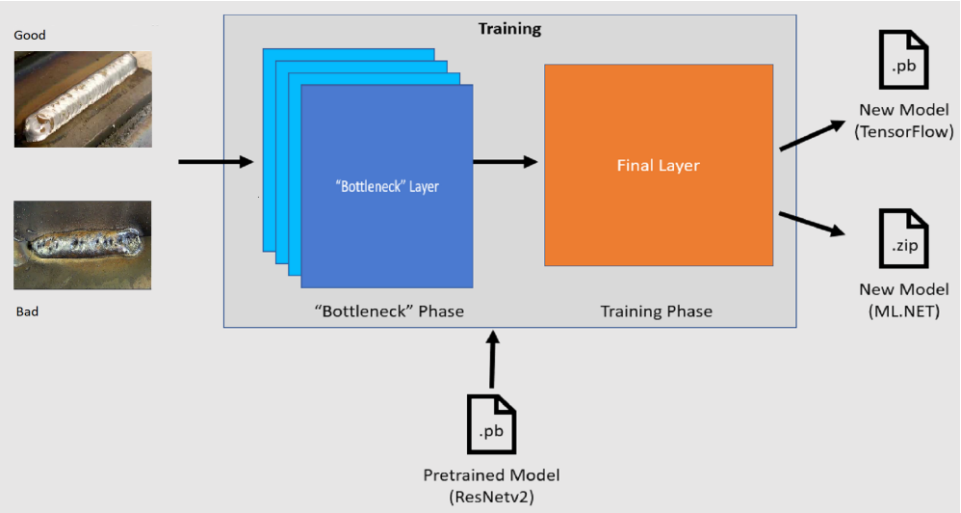

Figure 27 Image Classification API

Bottleneck Phase:

During the bottleneck phase, the set of training images is loaded and the pixel values are used as input, or features, for the frozen layers of the pre-trained model. The frozen layers include all of the layers in the neural network up to the penultimate layer, informally known as the bottleneck layer.

These layers are referred to as frozen because no training will occur on these layers and operations are pass-through. It's at these frozen layers where the lower-level patterns that help a model differentiate between the different classes are computed. The larger the number of layers, the more computationally intensive this step is.

Fortunately, since this is a one-time calculation, the results can be cached and used in later runs when experimenting with different parameters.

Training Phase:

Once the output values from the bottleneck phase are computed, they are used as input to retrain the final layer of the model. This process is iterative and runs for the number of times specified by model parameters.

During each run, the loss and accuracy are evaluated.
Then, the appropriate adjustments are made to improve the model with the goal of minimizing the loss and maximizing the accuracy.

Once training is finished, model formats are output .zip ML.NET serialized version of the model. When working in environments supported by ML.NET, it is recommended to use the .zip version of the model.

5.1.5 Evaluation

One of the most common techniques for model evaluation and model selection in machine learning practice is *K-fold cross-validation*.

The main idea behind cross-validation is that each observation in the dataset has the opportunity of being tested.
K-fold cross-validation is a special case of *cross-validation* where we iterate over a dataset set k times. In each round, dataset split into k parts: one part is used for validation, and the remaining k−1 parts are merged into a training subset for model evaluation. The figure below illustrates the process of 5-fold cross-validation:

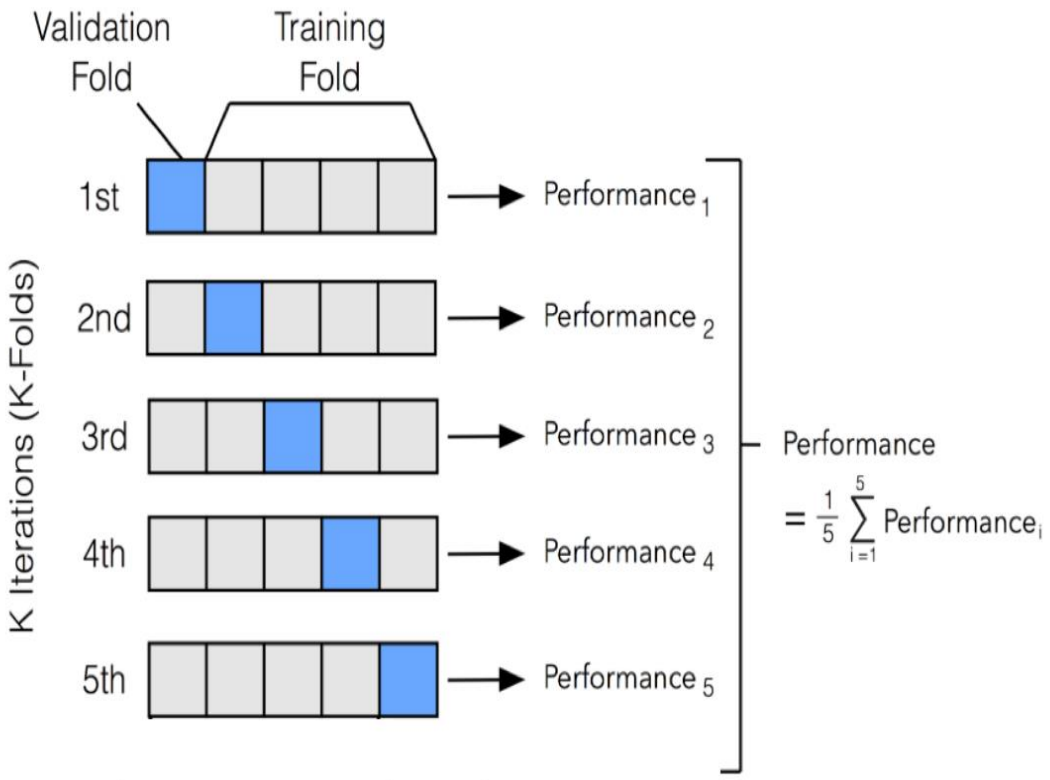

Figure 28 K-fold validation [25]

In 5-fold cross-validation, this procedure will result in 5 models fitted on distinct yet partly overlapping training sets and evaluated on non-overlapping validation sets.

Eventually, the cross-validation performance has been computed as the arithmetic means over the kk performance estimates from the validation sets. The main benefit behind this approach versus a simple train/test split is to reduce the pessimistic bias by

using more training data in contrast to setting aside a relatively large portion of the dataset as test data.

And here is the code of the *K-fold cross-validation* evaluation:

```
  private static void Evaluate(MLContext mlContext, IDataView trainingDataView,
IEstimator<ITransformer> trainingPipeline)
        {
            // Cross-Validate with single dataset (since we don't have two datasets, one for training
and for evaluate)
            // in order to evaluate and get the model's accuracy metrics
               Console.WriteLine("=============== Cross-validating to get model's accuracy metrics
               ===============");
            var crossValidationResults =
            mlContext.MulticlassClassification.CrossValidate(trainingDataView, trainingPipeline,
            numberOfFolds: 5, labelColumnName: "Label");
            PrintMulticlassClassificationFoldsAverageMetrics(crossValidationResults);
        }


/*-----------------------------------------*/

    public static void
PrintMulticlassClassificationFoldsAverageMetrics(IEnumerable<TrainCatalogBase.CrossValidationResult<Mu
lticlassClassificationMetrics>> crossValResults)
        {
            var metricsInMultipleFolds = crossValResults.Select(r => r.Metrics);

            var microAccuracyValues = metricsInMultipleFolds.Select(m => m.MicroAccuracy);
            var microAccuracyAverage = microAccuracyValues.Average();
            var microAccuraciesStdDeviation = CalculateStandardDeviation(microAccuracyValues);
            var microAccuraciesConfidenceInterval95 =
CalculateConfidenceInterval95(microAccuracyValues);

            var macroAccuracyValues = metricsInMultipleFolds.Select(m => m.MacroAccuracy);
            var macroAccuracyAverage = macroAccuracyValues.Average();
            var macroAccuraciesStdDeviation = CalculateStandardDeviation(macroAccuracyValues);
            var macroAccuraciesConfidenceInterval95 =
CalculateConfidenceInterval95(macroAccuracyValues);

            var logLossValues = metricsInMultipleFolds.Select(m => m.LogLoss);
            var logLossAverage = logLossValues.Average();
            var logLossStdDeviation = CalculateStandardDeviation(logLossValues);
            var logLossConfidenceInterval95 = CalculateConfidenceInterval95(logLossValues);

            var logLossReductionValues = metricsInMultipleFolds.Select(m => m.LogLossReduction);
            var logLossReductionAverage = logLossReductionValues.Average();
            var logLossReductionStdDeviation = CalculateStandardDeviation(logLossReductionValues);
            var logLossReductionConfidenceInterval95 =
CalculateConfidenceInterval95(logLossReductionValues);


Console.WriteLine($"*************************************************************************************
**************************");
            Console.WriteLine($"*       Metrics for Multi-class Classification model      ");
            Console.WriteLine($"*------------------------------------------------------------------------
---------------------------------------");
            Console.WriteLine($"*       Average MicroAccuracy:    {microAccuracyAverage:0.###}  -
Standard deviation: ({microAccuraciesStdDeviation:#.###})  - Confidence Interval 95%:
({microAccuraciesConfidenceInterval95:#.###})");
            Console.WriteLine($"*       Average MacroAccuracy:    {macroAccuracyAverage:0.###}  -
Standard deviation: ({macroAccuraciesStdDeviation:#.###})  - Confidence Interval 95%:
({macroAccuraciesConfidenceInterval95:#.###})");
            Console.WriteLine($"*       Average LogLoss:          {logLossAverage:#.###}  - Standard
deviation: ({logLossStdDeviation:#.###})  - Confidence Interval 95%:
({logLossConfidenceInterval95:#.###})");
            Console.WriteLine($"*       Average LogLossReduction: {logLossReductionAverage:#.###}  -
Standard deviation: ({logLossReductionStdDeviation:#.###})  - Confidence Interval 95%:
({logLossReductionConfidenceInterval95:#.###})");

Console.WriteLine($"*************************************************************************************
**************************");

        }
```

Code 4 Evaluation method

Evaluation metrics for Multi-Class Classification:

Metrics	Description	Look for
Micro-Accuracy	Micro-average Accuracy aggregates the contributions of all classes to compute the average metric. It is the fraction of instances predicted correctly. The micro-average does not take class membership into account. Basically, every sample-class pair contributes equally to the accuracy metric.	**The closer to 1.00, the better**. In a multi-class classification task, micro-accuracy is preferable over macro-accuracy if you suspect there might be class imbalance (i.e you may have many more examples of one class than of other classes).
Macro-Accuracy	Macro-average Accuracy is the average accuracy at the class level. The accuracy for each class is computed and the macro-accuracy is the average of these accuracies. Basically, every class contributes equally to the accuracy metric. Minority classes are given equal weight as the larger classes. The macro-average metric gives the same weight to each class, no matter how many instances from that class the dataset contains.	**The closer to 1.00, the better**. It computes the metric independently for each class and then takes the average (hence treating all classes equally)
Log-loss	Logarithmic loss measures the performance of a classification model where the prediction input is a probability value between 0.00 and 1.00. Log-loss increases as the predicted probability diverges from the actual label.	**The closer to 0.00, the better**. A perfect model would have a log-loss of 0.00. The goal of our machine learning models is to minimize this value.
Log-Loss Reduction	Logarithmic loss reduction can be interpreted as the advantage of the classifier over a random prediction.	**Ranges from -inf and 1.00, where 1.00 is perfect predictions and 0.00 indicates mean predictions**. For example, if the value equals 0.20, it can be interpreted as "the probability of a correct prediction is 20% better than random guessing"

Figure 29 Evaluation metrics for Multi-class Classification

And the evaluation results of this model using K-fold cross-validation were:

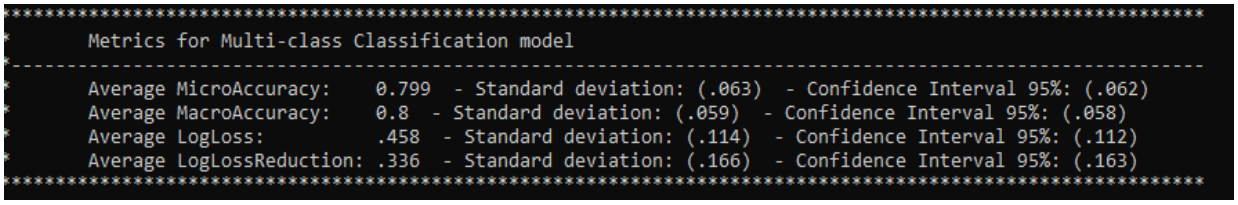

Figure 30 Metrics For Multi-Class Classification

Each value in Figure 30 has been explained in Figure 29, and Micro-Accuracy indicates the quality of a multiclass classification task which it's now around 0.8, this accuracy is accepted for now, and it should improve to be over 90%.

5.2 Using Model - Consuming

Consuming (Production *Mode*) is the step where the model will be loaded after it has been saved and used to predict the result from the input values.

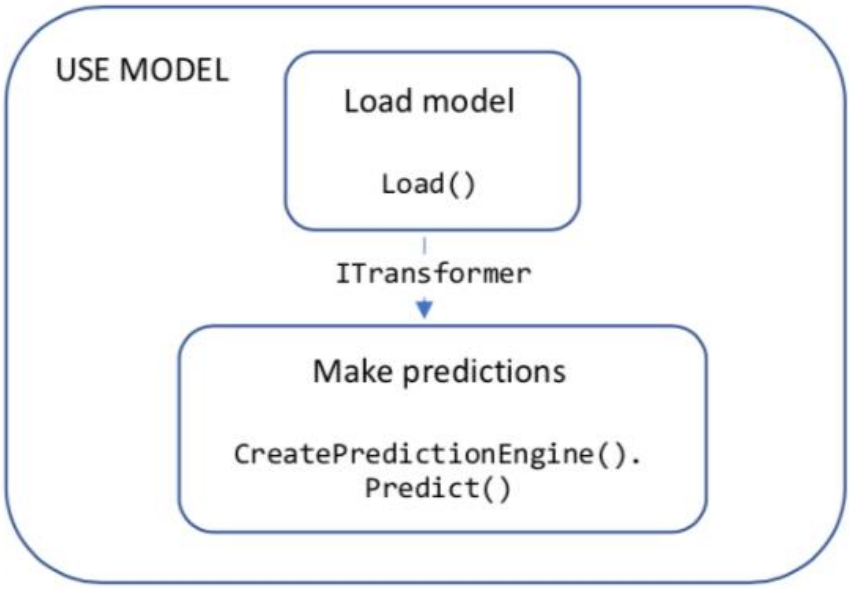

Figure 31 Using Model. [23]

When an image pass-through this model it will give the predicted score from 0 – 100 for each category, and since the categories have been adjusted to be only “Bad - Good” for now, then the predicted score will be 0-100% for Bad and 0-100% for good, where the percentage of the input images belong to each category.

The following code implements the loading and prediction procedures:

```
public  class ModelClass
{

   public  ConsumeModel consumeModel;

   public  void LoadModel(string modelpath)
        {
            if(consumeModel==null)
             consumeModel = new ConsumeModel(modelpath);

        }
   public List<KeyValuePair<string,string>> Predect(string FilePath, string modelpath)
        {
            LoadModel(modelpath);
       List<KeyValuePair<string, string>> result = new List<KeyValuePair<string,
       string>>();
            ModelInput sampleData = new ModelInput()
            {
                ImageSource = FilePath

            };
            var predictionResult = consumeModel.Predict(sampleData);
                   result.Add(new KeyValuePair<string, string>("ImageSource",
       sampleData.ImageSource));
            result.Add(new KeyValuePair<string, string>("Prediction",
       predictionResult.Prediction));
            result.Add(new KeyValuePair<string, string>("BadScore",
       predictionResult.Score[0].ToString()));
            result.Add(new KeyValuePair<string, string>("GoodScore",
       predictionResult.Score[1].ToString()));
            return result;
         }

}
```

Code 5 Model Class

The Consuming in this thesis has been limited to be used in an *MVC* website (pattern of *ASP.NET*), and *MVC* is an abbreviation of:

- **Model**: which represents the underlying, logical structure of data in a software application and the high-level class associated with it. This object model does not contain any information about the user interface.
- **View**: which is a collection of classes representing the elements in the user interface (all of the things the user can see and respond to on the screen, such as buttons, display boxes, and so forth).
- **Controller**: which represents the classes connecting the model and the view, and is used to communicate between classes in the model and view.

6 Results and Analysis

Since the project is for images analyzing, the website has been built to have a method to upload images and pass them to the model to analyze and give its score, and when the score has fetched, the project represents it by using the Amchart.js libraries, which is full of useful charts and here is one of the charts.

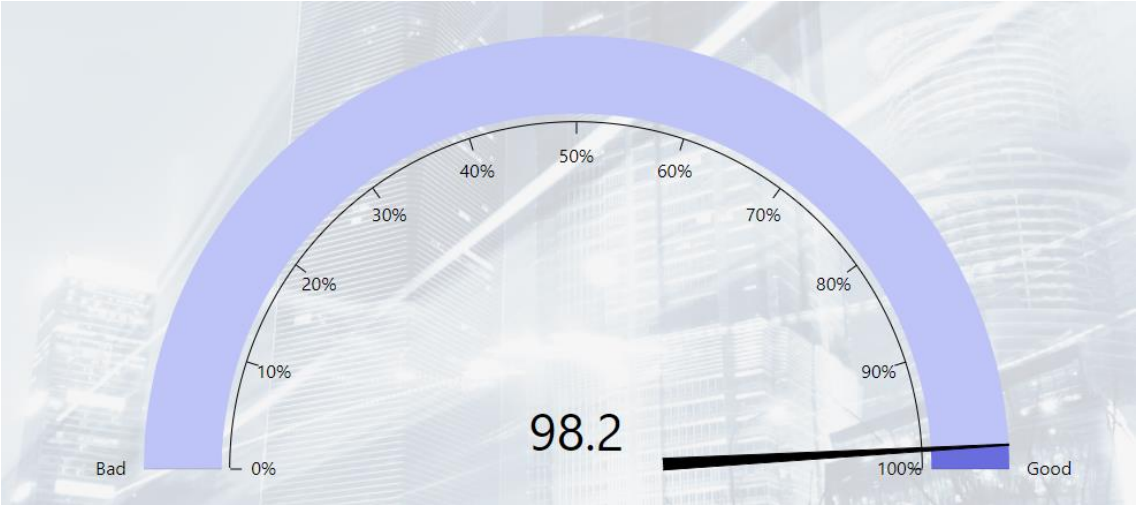

Figure 32 Gauge Chart

The gauge score in Figure (32) shows the expectation of the welding inside an image whether it's a good welding or a bad one, and in the previous example, the gauge shows that the probability of the welding is a good welding by 98.2% so it’s in the good welding class.

The previous chart will analyze one image, but also this project handles a series of images in case taking the average of welding quality is needed at once for many parts, and actually, this is how to quality ratio is taken (Figure 34).

The following Figure shows an example of testing one welding image, where the gauge is a chart that changes according to the result and the bottom left image is an example of how to take the picture.

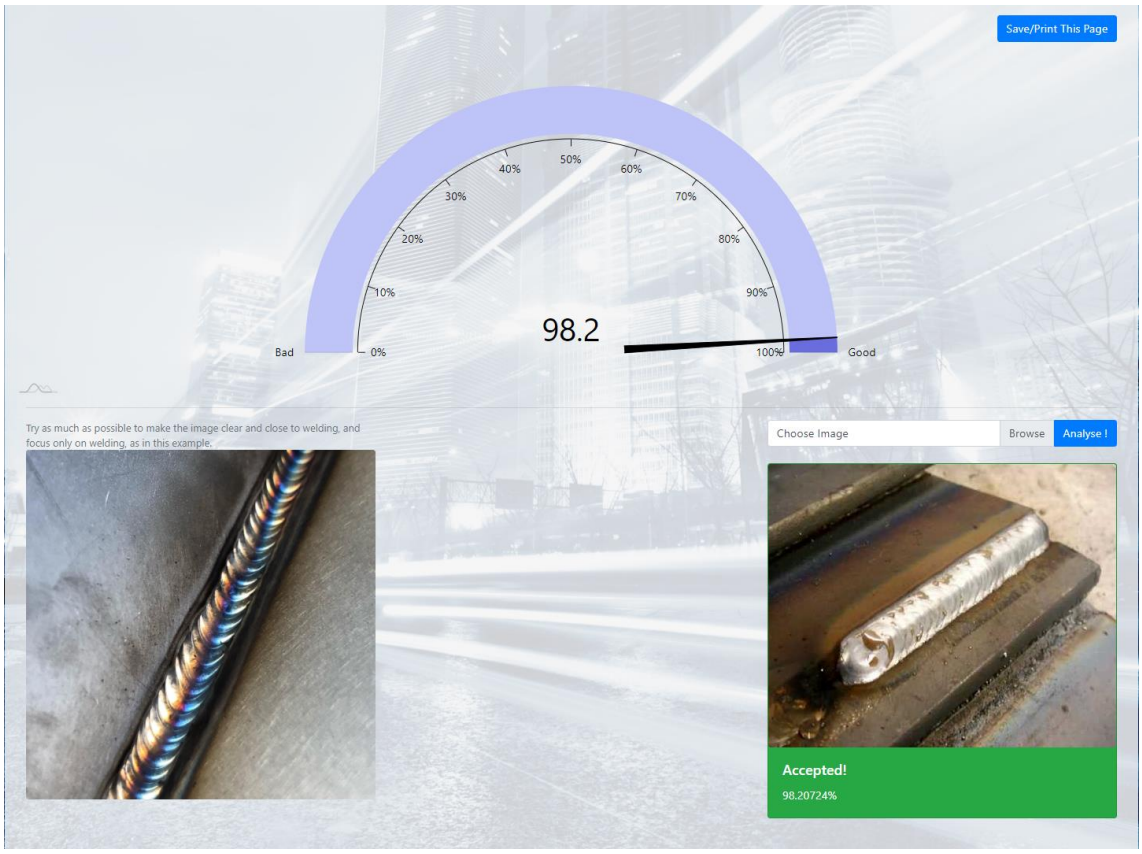

Figure 33 One Part Checking

The following Figure 34 shows a plot chart that plots the results, and a card gives the average value of the results (the red card) and a table shows the results individually:

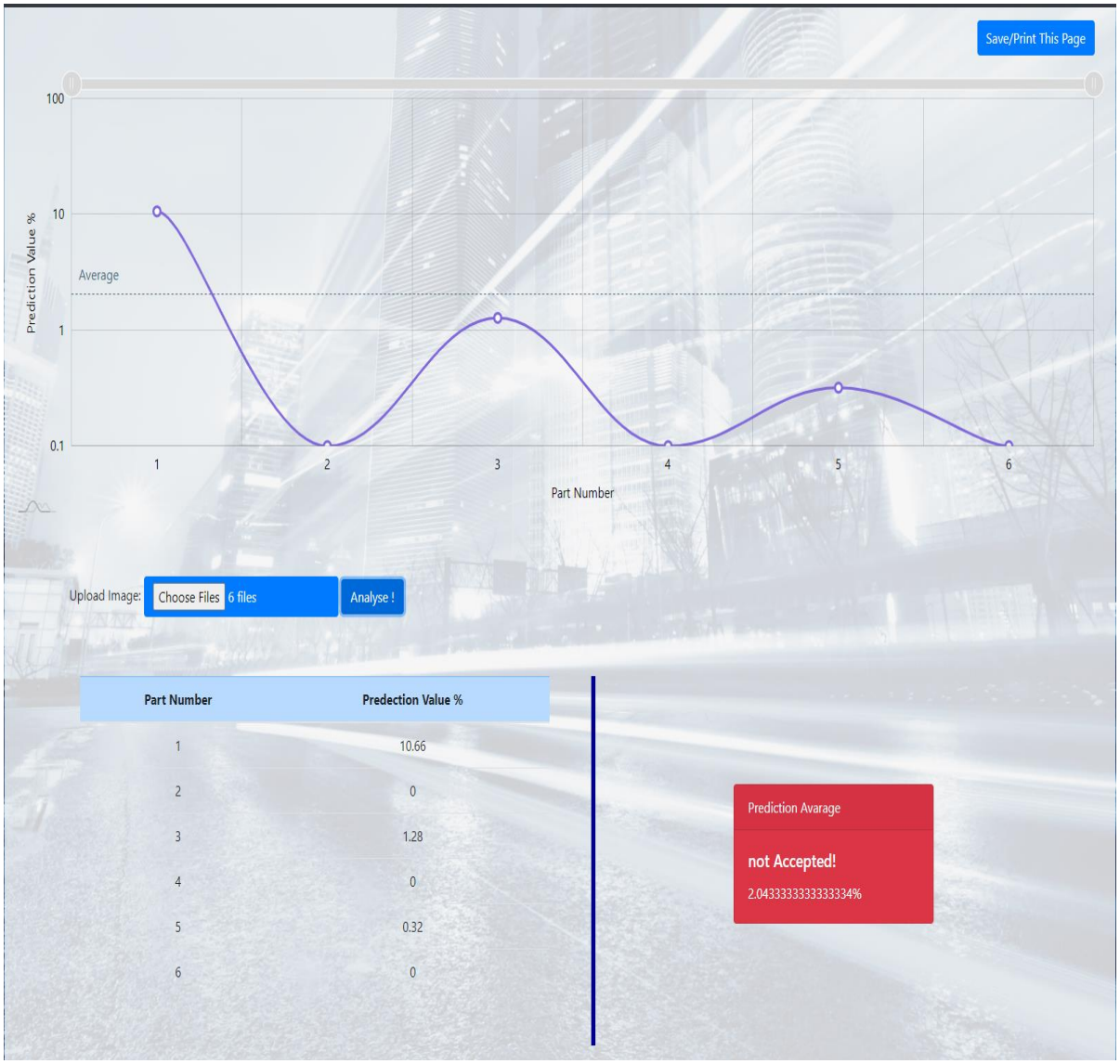

Figure 34 Many parts checking

The following figure shows the results by using the *Postman* application for testing the API where 1 image has been sent to be tested. The result is from 1 for each class, and the results show that the probability to be a good welding is 0.955 (about 96%), and to be bad one is 0.04 (4%), so it's a good welding therefore it's passed.

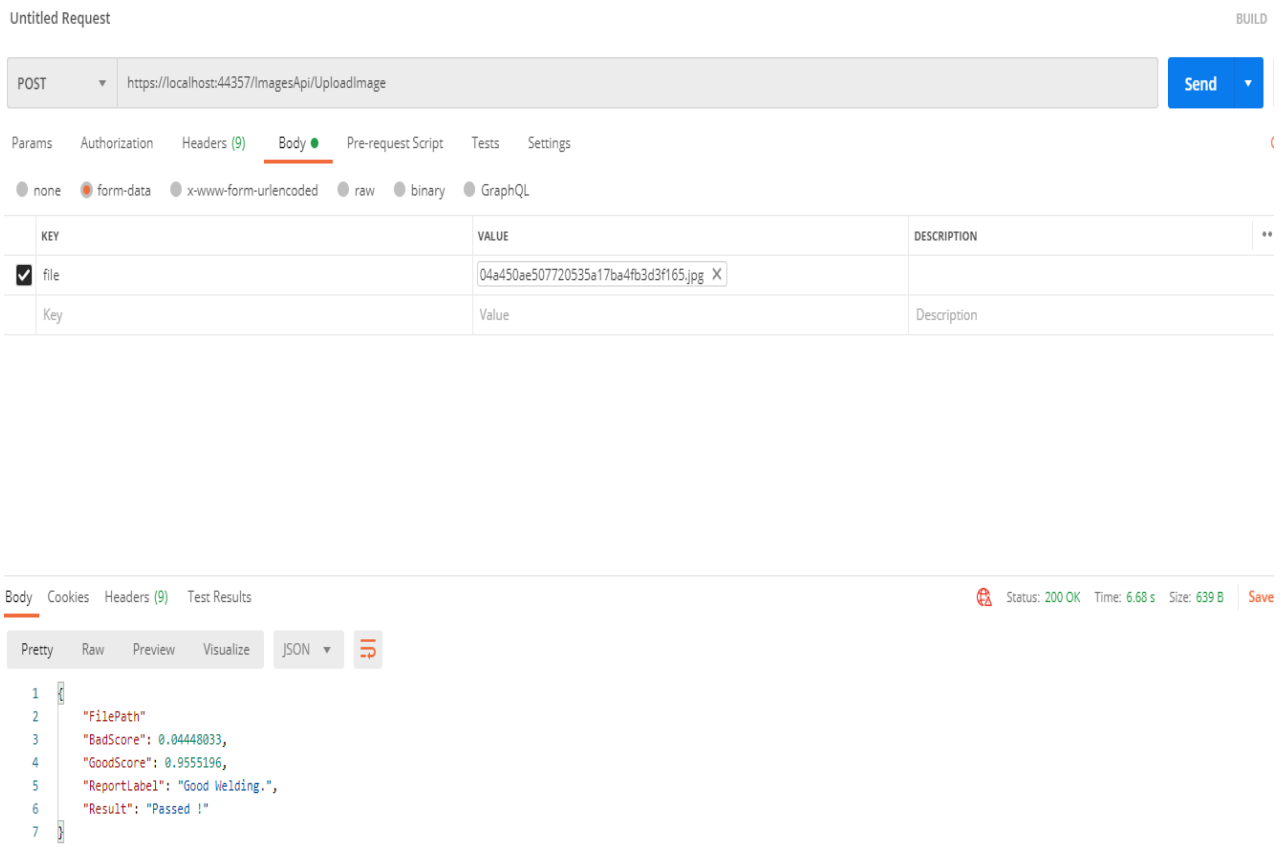

Figure 35 Postman Testing

And as what has been demonstrated in the Evaluation section accuracy of the model was accepted for this study since it gives the right scores and it will be improved later by adding more correct samples in the model building, and the decision of whether accuracy of the model accepted or no can be taken by comparing the results Figure 30 with the table in Figure 29, and by testing the consuming in the real life.

7 Discussion and Conclusions

AI image classification could solve the lack of professionals by giving the welder the quality results of the welding, and it could accelerate and improve the process quality. And the more it has samples in the dataset while building the model, the more precise results should come in the analysis, and this study proved that it is possible to create such a system (giving a report on whether welding is acceptable or not, i.e., good or bad) using AI, and how the ML model has been built and consumed and how accurate it was based on this study.

The model was tested and used to classify images, and good results were obtained, and it had an accuracy of 80% which is accepted for now, and it should be improve it be over 90% in the next development stages by having more samples.

It is important to note that this study will be expanded in the next stages in the company. And to develop this model to obtain more categories of welding quality, the developer must collect sufficient images for each needed category and put them in a separate folder labeled by the name of the new category (the new class) to be included in the model later. For example, if one wants to expand the model to classify the weldings whether they are have spots, cuts, or have holes, images must be collected for each of those conditions and put them separately in folders according to their classes. The images should be taken such a way that it shows the welding area as far away from other objects as possible. Also their resolutions must be moderate to avoid delays caused by high resolution images in the model building process, and also to avoid low resolution images where weldings are difficult to be recognized from the images. The images must also include all the sides of the weldings to avoid overfitting as what has been mentioned previously in this study.

Artificial intelligence and machine learning and their branches are like the oceans, and it is impossible to review all the algorithms and matters related to them in one thesis. Therefore, the focus was on the topics used in this project only, and how to create a machine learning model theoretically and programmatically, and how to use it to solve a concrete physical problem. And with a clear target, how to start programming and using artificial intelligence and machine learning in daily life, and this study will be expanded and developed to include the desired welding classifications in the company and to obtain a high accuracy of the model in the future.

And since most of the sciences and industries have become dependent in one way or another on artificial intelligence, it was necessary to take this initiative to solve a problem using this technology, and to highlight the idea of starting to use this intelligence in various ways.

As a conclusion, one can state the following:

A. This project has speeded up the process of checking the welding samples by giving an estimate of the weld quality, and thus avoiding the need to wait for an expert to examine welding samples.

B. The project was succeeded by reaching what was planned, and the most important things that succeeded in it are the following:
    a. The project succeeded in proving the theory of using artificial intelligence and machine language in image recognition according to what was taught in its model.
    b. It succeeded in differentiating between bad and good weldings.
    c. Giving the average percentage to the quality of several welding parts.

C. The project is expandable so other features and classes can be added when enough samples are collected for every case one needs to detect.

D. This project can open a lot of new ideas, that one can also make this kind of intelligence solve other things that belong to the quality (not just the welding) and other things.

References

[1] K. Jessica, “Artificial Intelligence System Analyzes Chest X-Rays in 10 Seconds,” www.healthitanalytics.com/, 2019.

[2] F. Fatemi, “Predicting Sales Success With AI,” www.forbes.com, 2018.

[3] “About,” Peikko.com.

[4] ARTSA, “Welding Quality,” www.artsa.com.au, 2017.

[5] ESAB, “Radiographic and Ultrasonic Testing of Welds,” www.esabna.com.

[6] A. Lab, “RADIOGRAPHIC WELD INSPECTIONS,” www.atslab.com, 2020.

[7] Eurogia, “Brokerage Webinar,” Eurogia, 2020.

[8] masterscan.com.

[9] H. University, “Elements Of AI,” www.elementsofai.com, 2020.

[10] C. Nicholson, “A Beginner's Guide to Neural Networks and Deep Learning,” wiki.pathmind.com.

[11] M. Tripathi, “Machine Learning Algorithm,” Datascience.foundation, 2019.

[12] J. Brownlee, “Difference Between Classification and Regression in Machine Learning,” www.machinelearningmastery.com, 2017.

[13] J. Leonel, “Supervised Learning,” www.medium.com, 2018.

[14] R. v. Loon, “Machine learning explained: Understanding supervised, unsupervised, and reinforcement learning,” www.bigdata-madesimple.com, 2018.

[15] A. Band, “multi-class-classification-one-vs-all-one-vs-one,” www.towardsdatascience.com, 2020.

[16] J.Brownlee,“one-vs-rest-and-one-vs-one-for-multi-class-classification,” www.machinelearningmastery.com, 2019.

[17] A. Haugen, “Introducing MLLib’s One-vs-rest Classifier,” www.antonhaugen.medium.com, 2020.

[18] Cesar, “Training Image Classification/Recognition models based on Deep Learning & Transfer Learning with ML.NET,” https://devblogs.microsoft.com/, 2019.

[19] R.Alake,“Machine Learning Image Classification With TensorFlow,” www.towardsdatascience.com, 2020.

[20] valohai, “machine learning pipeline,” www.valohai.com.

[21] V. Oy, “What Are the Benefits of a Machine Learning Pipeline?,” https://valohai.com/.

[22] A. SACHAN, “Tensorflow Tutorial 2: image classifier using convolutional neural network,” www.cv-tricks.com, 2018.

[23] Microsoft, “What is ML.NET and how does it work?,” www.docs.microsoft.com, 2021.

[24] Tensorflow, “Image classification,” Tensorflow, 2021.

[25] G. Öğündür, “K Fold Cross Validation,” https://medium.com/.