Czech University of Life Sciences Prague

Faculty of Economics and Management

Department of Information Engineering

Bachelor Thesis

Image recognition with ML.NET

Author: Mariia Redkoles
Supervisor: Ing. Petr Hanzlik, Ph.D.

CZECH UNIVERSITY OF LIFE SCIENCES PRAGUE

Faculty of Economics and Management

BACHELOR THESIS ASSIGNMENT

Mariia Redkoles

Systems Engineering and Informatics
Informatics

Thesis title

Image recognition with ML.NET

Objectives of thesis

The objective of this bachelor thesis is to create an application for image recognition. The intelligent agent will be implemented by generating an ML.NET image classification model from a pre-trained TensorFlow model. The agent will be wrapped in a simple C# application.

Methodology

The methodology of the thesis is based on analysis of technical and scientific sources focusing on artificial intelligence, machine learning, and object recognition. Based on the synthesis of the knowledge gained, a prototype C# application will be implemented to recognize objects in an image. The TensorFlow inception model's abilities are going to be used in ML.NET image classifier. The performance of the application will be assessed in terms of detection precision and classification accuracy.

The proposed extent of the thesis

40-50 pages

Keywords

Deep learning, Machine learning, Image recognition, Image classification, TensorFlow, ML.NET

Recommended information sources

CAPELLMAN, Jarred, 2020. Hands-On Machine Learning with ML.NET. Birmingham: Packt Publishing. ISBN 9781789801781.

CYGANEK, Boguslaw, 2013. Object Detection and Recognition in Digital Images: Theory and Practice. Chichester, UK: John Wiley and Sons. ISBN 978-0470976371.

PRICE, Mark J., 2019. C# 8.0 and .NET Core 3.0: Modern Cross-Platform Development: Build applications with C#, .NET Core, Entity Framework Core, ASP.NET Core, and ML.NET using Visual Studio Code. 4. Birmingham: Packt Publishing. ISBN 978-1788478120.

Expected date of thesis defence

2019/20 SS – FEM

The Bachelor Thesis Supervisor

Ing. Petr Hanzlík, Ph.D.

Supervising department

Department of Information Engineering

Electronic approval: 23. 2. 2021

Ing. Martin Pelikán, Ph.D.

Head of department

Electronic approval: 23. 2. 2021

Ing. Martin Pelikán, Ph.D.

Dean

Prague on 21. 08. 2021

Declaration

I declare that I have worked on my bachelor thesis titled "Image Recognition with ML.NET" by myself and I have used only the sources mentioned at the end of the thesis. As the author of the bachelor thesis, I declare that the thesis does not break copyrights of any person.

In Prague on 26.08.2021 _____

Acknowledgment

I would like to thank my supervisor Ing. Petr Hanzlík, Ph.D. for the time he spent helping me throughout the whole thesis, mentoring, great advice and ideas.

Image recognition with ML.NET

Abstract

This thesis is focused on developing software in order to make recycling easier, therefore more used. It will be implemented by using modern technologies such as Machine Learning, TensorFlow ML.NET, and others.

These technologies will be explained in great detail in the first part of the thesis, what they are, how they work, what they are made of, and what is their purpose. The second part, which is a practical part, will be about the step-by-step implementation and realization of the idea. As a result, a functional model for waste separation will be created. The model, after inserting an image of a waste product, will predict which category it belongs to.

Keywords: Deep learning, Machine learning, Image recognition, Image classification, TensorFlow, ML.NET

Klasifikace obrázků za pomocí ML.NET

Abstrakt

Tato práce se soustředí na zjednodušení recyklace, za použití strojového učení, TensorFlow, ML.NET, a dalších technologií.
Detaily (např.: jak fungují, z čeho se skládají, a k čemu se používají) těchto technologií budou popsány v první části této práce.
Druhá část, která je praktická, se bude týkat postupných kroků implementace zmíněných technologií do programu určeného ke třídění odpadu.
Finální program za pomoci netrénovaného modelu dokáže z fotky rozeznat do jaké kategorie daný kus odpadu patří.

Klíčová slova: Deep learning, Machine learning, Image recognition, Image classification, TensorFlow, ML.NET

Table of Content

List of figures

List of abbreviations

AI - Artificial Intelligence
ANI - Artificial Narrow Intelligence
AGI - Artificial General Intelligence
ASI - Artificial Super Intelligence
ASR - Automatic Speech Recognition
NLP - Natural Language Processing
CNN - Convolutional Neural Network
GDPR - General Data Protection Regulation
MIT - Massachusetts Institute of Technology
ReLU - Rectified Linear Unit
GELU - Gaussian Error Linear Unit
SELU - Scaled Exponential Linear Unit
NLP - Natural Language Processing
API - Application Programming Interface
L-BFGS algorithm - Limited-memory Broyden–Fletcher–Goldfarb–Shanno

1. Introduction

Humanity is facing one of the greatest problems in our history - pollution. It is not only causing irreparable damage to our planet(especially oceans) but also to us and our health. Speaking numbers about 40% of human deaths are caused by air, water, and soil pollution. In order to prevent the death of everything alive, it is time to face the problem and act immediately.

Unmanaged landfills have a severe impact on our climate. When organic waste products are decomposing they produce around 60% of methane, methane emissions are what causing global warming that can make our planet inhabitable. Alongside, landfills pollute oceans when water goes through them, it picks up lots of toxic substances.

Not only do landfills pollute the oceans, but we also release waste into water bodies, by this, we make it unsafe not only for humans but it also depletes aquatic ecosystems and kills its inhabitants.

Here are some of the benefits of garbage separation:

- We create lots of waste and with every year the number is increasing rapidly, so there is a need for a place to bury it, but what if in a couple of hundred years there is no more place? Recycling prevents this from happening, therefore this land can be used for better purposes such as growing trees and helping our environment;

- Waste decomposition is poisoning soil and water as well as causing global warming. Most part of the waste products can be reused, therefore decreasing the deadly impact;

- Production is using lots of natural resources such as water, wood, etc. By recycling, we don't need as much of it. For example, metals can be reused an unlimited amount of times, paper - up to 6, and plastic - one or two;

- Recycling and reusing are cheaper, considering the transportation, payments, electricity bills, etc. For example, aluminum cans when recycled, require 95% less energy than to produce new ones;

Considering aforesaid I have decided to help to resolve this problem. We live in an era of technology and automation is a great benefit since we can not make everyone care. In this thesis, we will try to create a model that will be able to separate garbage into three categories: paper, glass, and plastic. We will explore what stands behind this, the logic of it, and the technologies used.

2. Objectives and Methodology

2.1. Objectives

The objective of this bachelor thesis is to create an application for image recognition. The intelligent agent will be implemented by generating an ML.NET image classification model from a pre-trained TensorFlow model. The agent will be wrapped in a simple C# application.

2.2. Methodology

The methodology of the thesis is based on analysis of technical and scientific sources focusing on artificial intelligence, machine learning, and object recognition. Based on the synthesis of the knowledge gained, a prototype C# application will be implemented to recognize objects in an image. The TensorFlow Inception model's abilities are going to be used in an ML.NET image classifier. The performance of the application will be assessed in terms of detection precision and classification accuracy.

3. Literature Review

3.1 Artificial Intelligence

There is an eminent quote which I think is good to define AI - "It is the science and engineering of making intelligent machines, especially intelligent computer programs. It is related to the similar task of using computers to understand human intelligence, but AI does not have to confine itself to methods that are biologically observable." (McCarthy, 2004)

In other words, AI is a field of computer science where machines are trying to simulate human intelligence for problem-solving purposes based on input data, also it is the most complex of human creations yet.

There are two generalized types of AI nowadays, these are weak AI and strong AI.

Weak AI or it can also be called Artificial Narrow Intelligence (ANI) is pretty much all the AI used around us, even the most complex ones created by now fall under ANI. Weak AI is used to solve some particular tasks by using human-like capabilities, but nothing more than it is programmed to do. It is used for example by Tesla, Amazon, Apple, etc.

Strong AI is divided into two branches: Artificial General Intelligence (AGI) and Artificial Super Intelligence (ASI).

AGI is an idea of AI having human intelligence with an ability to make its own decisions, multi-task, learn, build plans, have its own consciousness, and pretty much function like a human, whereas ASI is an idea of AI being superior to human beings.

Artificial Intelligence has three sub-fields that are widely used: Machine Learning, Deep Learning and Neural Networks. (IBM Cloud Education, 27 May 2020)

3.1.1 Types of AI

- **Reactive Machines** - the oldest as well as the most primitive type of AI. These machines cannot create their own memories or use past experiences, which means they cannot "learn". Reactive Machines can only be used for responding to a set of inputs in other words they act to what they see. There is a well-known example of this type of machine, which is IBM's Deep Blue.

- **Limited memory** - as distinct from reactive machines, this technology can look into the past. It means that it gets the knowledge from historical data, previously learned information, events, etc., and it analyzes its actions for making a better decision. This technology is used nearly everywhere where AI is: self-driving cars, chatbots, virtual voice assistants, etc.

- **Theory of mind** - in psychology, it means understanding that other agents or entities in this world can have emotions, needs, thoughts, or beliefs that affect their behavior. Unlike previous technology, it is a work in progress and does not exist yet, but the essential idea of it is understanding humans as individuals with social interactions, motives, and expectations how to be treated - therefore act according to it.

- **Self-aware** - this type of AI exists so far only hypothetically. An objective is to create a technology that will be self-aware or have consciousness. It means that it will have its own emotions, needs, and desires as well as possibly self-preservation ideas.

3.1.2 AI in use

Nowadays there are many for AI technology. The most common are following:

- **Automatic Speech Recognition (ASR)** - is a technology that is using Natural Language Processing (NLP) to convert spoken language into text. Siri is a noteworthy example of it.

- **Computer Vision** - using Convolutional Neural Networks (CNN) this technology allows a computer to derive meaningful information from images, videos, or some other inputs for further actions. For example, Tesla has 8 cameras in their car that transform all the incoming data into one vector space for a full vision.

- **Customer Service** - this technology came in handy, no need to hire more people since virtual agents or bots can do the job. Virtual agents are created to answer the most frequently asked questions. Telegram is famous for its bots.

- **Automated Stock Trading** - there are server-based trading platforms that can perform trading without human intervention.

- **Recommendation Engines** - using past behavioral data, this technology provides recommendations, for example in online shopping, when on checkout it offers a customer other products based on his interests or Netflix's recommendations.

3.2 Machine Learning

ML is a branch of AI which focuses on letting machines learn for themselves on some set of data or past experiences without being fully programmed. The main objective is that they perform the task with just an observation or instructions, which in a sense is an imitation of the way humans learn.

Though training a model normally takes a long time since it is trained over vast quantities of data - results are generally more efficient and accurate. Also, there is an option of combining it with some other technologies - results can outgrow

themselves. For example application of ML in IoT has brought us to creating robotic vacuum cleaners, smart thermostat, etc.

3.2.1 Machine Learning Methods

Machine learning algorithms are normally distinguished into:

- **Supervised** - these methods use a pre-labeled training dataset for future predictions. When data is fed to a model it is adjusting its weights to learn the situation, then it creates a mapping function that will make predictions about the output values. Furthermore, this model is able to compare its outcome to the right one and learn from it. The goal is to achieve an acceptable level of performance.

- **Unsupervised** - it is called unsupervised since there is no human intervention, a model gets only an unlabeled input dataset. After that, it uses machine learning algorithms to find hidden patterns, differences, or similarities in order to learn more about the data. The model is left on its own, there is no right or wrong answer.

- **Semi-supervised** - is a golden mean between supervised and unsupervised learning. Semi-supervised learning uses a small amount of labeled data besides unlabeled to guide the classification through the unlabeled dataset. Also this method can be used when there is not enough labeled data for supervised learning. In a way, it is more efficient to use semi-supervised learning since it is very time-consuming and expensive to label the data, whereas the unlabeled data can have free access.

- **Reinforcement** - this machine learning method is similar to supervised learning, except there is no training dataset. The model is put in a game-like environment where the goal is to maximize achieved points. Without any guidance or hints, it must make a sequence of decisions in order to get a reward, otherwise there is a penalty. It is a trial-and-error process, but once

it's done, the result is a very sophisticated tactic to solve the problem. A good example is IBM's Watson that used reinforced learning to win "Jeopardy!" in 2011. ("IBM Watson: the inside story of how the Jeopardy-winning supercomputer was born, and what it wants to do next", Jo Best, September 9,2013)

Here is a more detailed overview of what they can do:

- **Classification** - predicts a category for an input variable, for example: "apple" or "pear", "disease" or "no disease" etc.

- **Regression** - helps to predict a numerical output value such as money or price based on given data with similar properties.

- **Clustering** - is when you need to find some inherent groupings in the data with similar characteristics, for example separating customers into groups just by their shopping behavior. In clustering, there is no output data used for training, instead the algorithm itself decides what is the output.

- **Association** - this problem is to analyze a vast amount of data to find hidden patterns that describe the given data. For example, what group of people that buy product X, also tend to get product Y.

- **Dimensionality reduction** - name defines itself, dimensionality reduction is used to remove least important data that can have thousands of features not needed for analysis.

- **Ensemble** - this method is a merge of other methods in a way, it combines them. As a result, it can get higher-quality predictions than each one of them by itself.

- **Word embeddings** - they capture the meaning of the word as well as find similarities between, and then allow us to do arithmetic with them.

3.2.2 How it works

We can estimate three main parts of a machine learning process:

- **A decision process** - this step is about creating an estimate of patterns in a given dataset, no matter labeled or unlabeled.

- **An error function** - this method estimates how good the prediction is by comparing it to already known examples (if such exist).

- **An updating or optimization process** - at this stage the model learns and adjusts weights better to prevent the same mistake and improve overall.

Following figure represents a more detailed overview of the steps in machine learning.

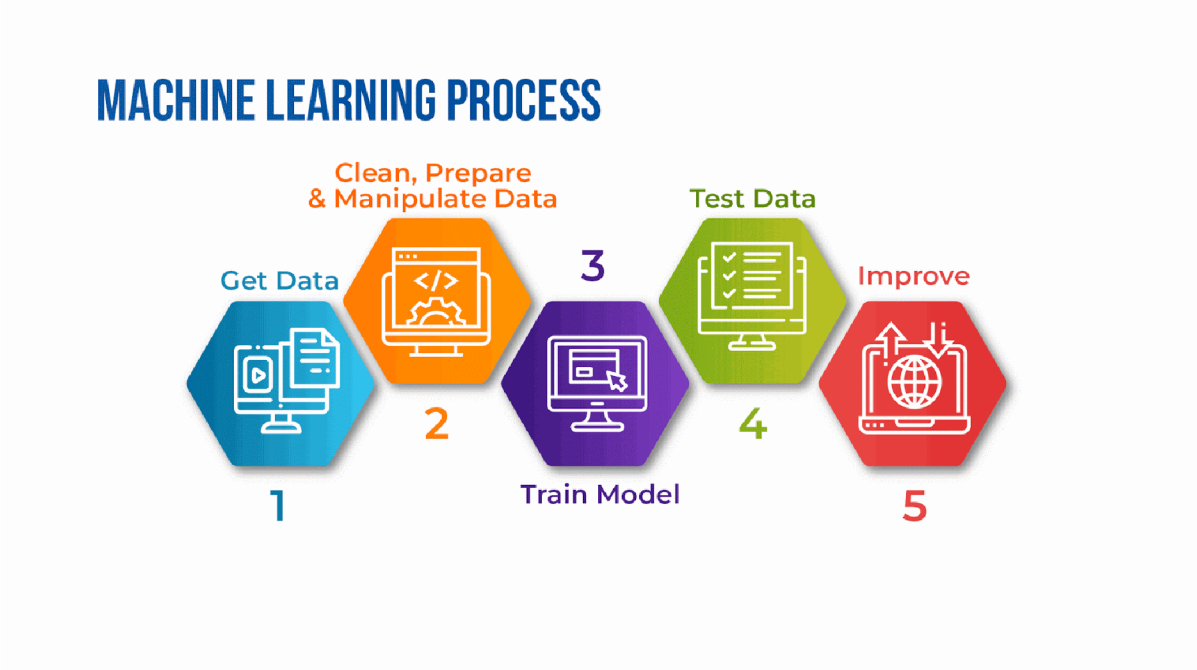

Figure 1. Machine Learning Process shown in the image (Adnan Sheikh, 2021)

3.2.3 Challenges of ML

AI has not only brought us lots of help but also a lot of ethical concerns. And here is a detailed overview of some of them:

- **Technological Singularity** - it is a very often discussed topic, should we use and trust autonomous systems such as self-driving cars or we must only create semi-autonomous vehicles and trust only ourselves with our lives? Can these cars fulfill their purpose, keep the driver safe and follow regulations designed for everyone or should we be more concerned about the human factor of the rest of the drivers and stay always aware? These and other similar questions haven't found an answer yet in our society. One can be said for sure, AI must earn our trust, but is it even going to happen?

- **AI Impact on Jobs** - there is definitely a lot of noise around this topic. It only depends on the perspective you look at it. As with any other innovation, the demand will be shifted to some other area as well as new positions will be created.

- **Accountability** - another big impact on the fear of AI is that there is no legislation for it. Yes, there are the three laws of robotics and ethicists are working aside with researchers, but can anyone guarantee safety?

- **Privacy** - data collection has rapidly increased in the past decade, making people concerned about their personal information. Because of that, governments have invented new regulations for companies, such as GDPR in Europe that gives people more control over their data.

- **Bias and Discrimination** - there are multiple questions regarding the ethics of AI, will there be discrimination? Amazon has tried it out and implemented AI in an attempt to simplify the hiring process. It has been called out immediately since the results were unacceptable, AI has biased job candidates by gender for any technical positions, which brings us to the point of making better

choices when choosing what data should be used in the hiring process. And it is only one of the cases, more and more are coming before it will be made right.

3.2.4. Limitations of ML

Machine learning is a very powerful technology that can make a business grow with rocket speed. But before using it, it is always better to be aware of its limitations, which are following:

- **Insufficient, unknown, or excessive data** - either one of these errors can occur during the training process since data is the most important qualifier. Predictive algorithms won't be able to learn in time, results can be far from being true, these problems may occur if unsuitable data is being fed to a model, or when excessive amounts are being added, so the model hasn't been trained to process that amount of new parameters simultaneously. Also, every single model must be trained differently and on a different dataset, since requirements are always various.

- **Training time** - to get a reliable, and accurate model it is very important to train it well, try out different datasets, check and correct its predictions to improve the model. The software learns only after a certain period of time, and there are multiple training rounds, so the model must be corrected quite often in order to improve accuracy and keep the reliability score below what is permissible.

- **Approximation of results** - after all these years, ML has improved, still, it cannot give a 100% accurate answer, and there is still a place for error or uncertainty. Anyways, it is still more beneficial to have at least some answer even if it is with an error than no answer at all. Therefore learn to work and understand these types of errors, for example, to prevent a model from being biased.

- **Data Interpretation** - none of the algorithms available cannot make the right prediction due to one reason, it doesn't understand the meaning of the data and the task given. That is why there is still a need for humans to be involved.

3.2.5 Use Cases for ML

Machine learning is used in multiple areas and for different purposes. These are only some of them:

- **Customer service** - this area is full of chatbots, which are more efficient to answer the most frequently asked questions (FAQ), such as troubleshooting, delivery, recommendations, etc. Examples are on most of the commercial websites, Facebook Messenger, Vkontakte, Telegram, and others.

- **Recommendation engines** - AI that is using customers' previous consumption data to create personalized recommendations, for online shopping at the checkout, for example.

- **Computer vision** - this AI technology is using visual inputs, analyzes them, and makes an action. For example, Tesla is using multiple cameras to recreate the surrounding to a dot map in vector space, for their self-driving cars.

- **Automatic speech recognition (ASR)** - is the use of NLP to process human speech into a written text. Used in most modern devices not only for voice typing but also for simple tasks like voice search, setting an alarm, or calling someone. Siri, Cortana, and other virtual assistants use it.

3.3 Neural Networks

3.3.1 History

Neural Networks were first created in the distant 1944 by Warren McCullough and Walter Pitts, who later became founding members of the so-called "first cognitive science department" at MIT. The research was going on and off for many years until the 1980s when it was fueled by a new increased power of graphical chips.

When it was first created it was far from the way it is now, it had thresholds and weights with no layers and no training mechanism(specified by its creators). The idea behind this was that our brain could be like a computing device, since their model could compute any function as a computer could, which is indeed valuable for neuroscientific research, for example, to understand how the brain processes information.

Perceptron is the first trainable NN presented in 1957 by a psychologist of Cornell University - Franc Rosenblatt. It was finally looking more or less like a modern NN except it has only one layer.

“Of course, all of these limitations kind of disappear if you take machinery that is a little more complicated — like, two layers” - said Tomasso Poggio, professor of brain and cognitive science at MIT.

And yes, by the 1980s, layers were added and NN has experienced a renaissance, but then another problem has appeared, how to understand it? Even nowadays we still don't know the answer, but we have adopted and deciphered some of its analytic strategies.

Today, modern models can have up to 50 layers which is a nice improvement compared to day one, and it is considered the best performing system in almost all of the fields of AI.

3.3.2 Artificial Neural Networks

Neural networks are circuits of neurons, therefore exist in two types - Biological Neural Networks and Artificial Neural Networks. Biological Neural Networks are made of biological neurons found in our brain, which brings us to

Artificial Neural Networks that are inspired by them - made of artificial neurons or nodes in order to solve AI tasks, they imitate some of the basic functionalities of our brain but in a simplified way. Or else, it is a set of algorithms that is to find underlying connections in a dataset in a process that imitates a process in a human brain.

A typical Neural Network consists of input, output, and hidden layers.

An input layer provides us with data that the network is going to learn from, and an output layer provides us with a result of the whole process, and hidden layers are where all the mathematical operations are happening.

Connections between neurons in ANN are presented as weights between nodes(numerical value) and they are a representation of an axon in our brain. There are excitatory connections - for positive weights, and inhibitory connections - for negative weights.

When the model learns, it changes these weights in order to solve a given task, it must get a certain weight for it to be done, but no one can predict what the value would be before it actually is done. Weights as datasets are different for each task and the model has to learn them.

Given inputs are modified in weights and being summed, which creates a linear combination, then goes to an activation function and gives an output. The range of output can vary, from 0 to 1, or from -1 to 1.

ANN can derive information and form a conclusion from information that is very complex and unrelated at a first glance.

The flow of the information in NN can be of two types:

- **Feedforward propagation** - data moves only forwards, so the activation function makes a "gate" in a way.

- **Backpropagation** - in this case, weights of the connections are being adjusted, therefore we minimize the difference in the outcome of an obtained outcome vector and a desired one.

Following image describes the architecture of an ANN.

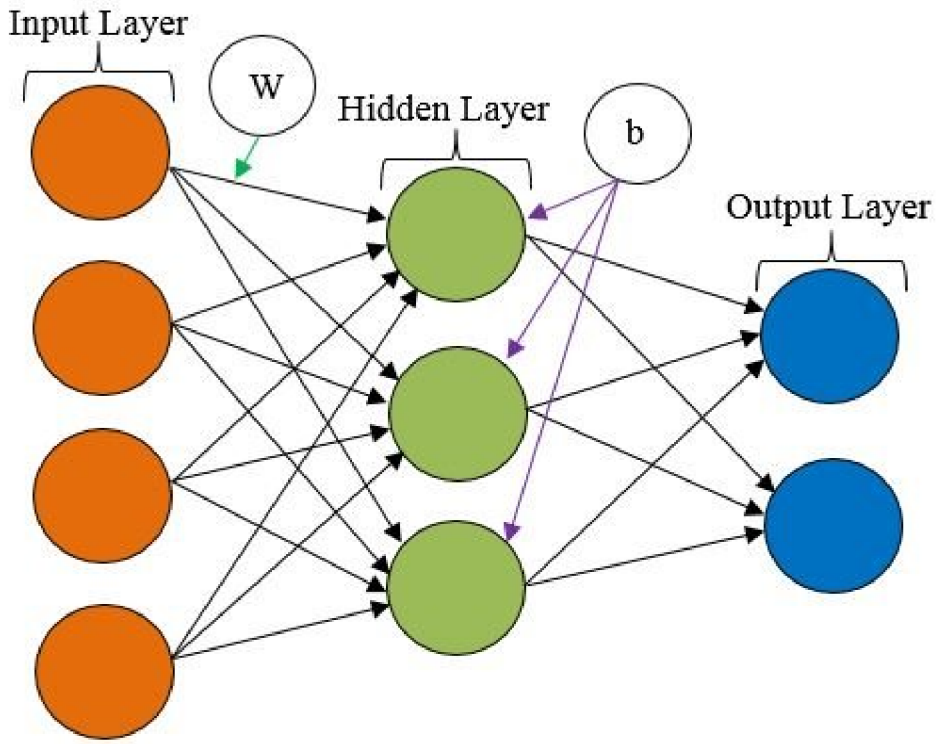

Figure 2. *The basic ANN architecture* by Akizur Rahman (2019) shows components of ANN

3.3.3 How does the model learn?

In the picture given above, we can see an input layer made, in our case, of two neurons(nodes) and is connected with nodes from the next layer using weights; these weights are assigned to each neuron according to the relative importance that these neurons have against other inputs.

In the first step, all the nodes' values, as well as weights', are multiplied and summarized, they form values of the next layer, and based on this value, a predefined activation function will decide if the node is going to be activated in the next layer or not. Node being activated or not decides whether the node's input is important in the prediction process.

After we get a prediction during the frontpropagation we then compare them to actual ones using the Loss Function. A smaller loss value means a more accurate prediction, so if we minimize the loss function we thereby improve our model.

In order to minimize it, we must find a set of values of the weights for which the value is as small as possible. To achieve it, a different optimization function must be used, called - Gradient Descent.

During this method a derivative of a loss function is used, the so-called "gradient", to adjust the weights according to the error they have caused.

The weights are being repeatedly updated until they reach their optimal, needed value.

After going through all these layers, calculations, and activations - the result is forwarded to an output layer and can finally be seen.

3.3.4 Types of ANN

ANNs are normally described by the number of layers that show their depth, hidden layers describe depth, it explains why the term neural networks is used as a synonym to deep learning. Else, it can be described by the number of inputs and outputs each node has or the number of hidden nodes.

These are the types that could be selected as most specific:

- **Feed-forward Neural Networks** - in this model, in order to get to the output node, information is passed in one direction and might not have any hidden layers, therefore making it easier to interpret. It is one of the simplest types and is often used for computer vision or facial recognition.

- **Recurrent Neural Networks** - a more complex type, here model is being refed with its previous results of the processing nodes to learn to predict the outcome of the layer. It starts the same, with frontpropagation, but then this memory node remembers the previous result and reuses it in case if the prediction is incorrect, therefore self-improves and moves towards the right result during the backpropagation. Most often this type of ANN is used in text-to-speech.

- **Convolutional Neural Networks** - one of the most popular technologies nowadays, widely used in image recognition, facial recognition, NLP, text digitalization, etc. This model contains one or more convolutional layers that could be either pooled or connected and uses a variation of multilayer perceptrons. Convolutional layers pick the area of the picture with the biggest amount of broken rectangles and record it into feature maps, then send them out.

- **Deconvolutional Neural Networks** - this model is to find lost features that were before considered unimportant therefore it reverses a CNN model process. It is used in image analysis.

- **Modular Neural Networks** - it is a combination of neural networks independent from each other. It makes complex computational processes more efficient.

3.3.5 Advantages and disadvantages of ANN

Advantages of ANN include:

- ANN's can predict an outcome of unseen data since it works to find these unseen relationships;

- there is not just a database, information is being stored all over the network;

- ANN's are able to learn from events and make their own decisions only based on observations;

- a network is multi-tasking, meaning it can parallel process and do a few jobs at a time;

- no restrictions are applied to an input data;

- an ability to produce an outcome without having complete knowledge or with a loss of performance just based on the importance of the missing information;

- it is fault-tolerant, which means that when some cells are corrupted - ANN will still provide an outcome.

Disadvantages include:

- it can be hardware dependant because the ability of parallel processing requires particular processors;

- all the input data must be transformed into numerical in order to be fed to a model;

- the lack of explanations of why and how;

3.3.6 Types of Activation Functions

The use of Activation Functions is to show a non-linearity in the network. Normally, all the hidden layers use the same Activation Function within the same network.

The activation function defines the output that a node will generate, based upon its input:

- **Sigmoid Function** - is widely used, very well results are shown when we have to predict probabilities since the probability range is the same as the range of the output value of the function, and is from 0 to 1. A limitation is the

Vanishing Gradient problem, when more layers are used it makes the gradient too small to work effectively;

- **Tanh Function** - is similar to the previous function, except the range is from -1 to 1, it even has the same S-shape. The output of the function is zero centered, which means that we can separate the outcome values into positive, negative, or neutral, it is good for centering data and makes the learning process easier for further layers; has the same limitation as to the sigmoid function; Should not be used in hidden layers as well as Sigmoid Function due to a Vanishing Gradient problem;

- **Rectified Linear Unit (ReLU)** - is a derivative function that is more multi-tasking, it allows the backpropagation to work simultaneously; ReLU only deactivates nodes when the value goes below zero and does not activate them at once, it makes it more efficient than previous functions. A limitation faced by this function is called the "Dying ReLU Problem" - negative input values become zero instantly which decreases the ability to train well, by this - dead neurons are created, that won't ever get activated; Should only be used in hidden layers;

- **Leaky ReLU Function** - an updated version that solves the dying ReLU problem. Is more time-consuming because of the modification, in which the gradient for a negative value is a small number.

- **Softmax Function** - is mostly used in the last layer for multi-classifications, and is described as a combination of multiple sigmoids. It calculates relative probabilities and after returns a probability for each of the classes;

- **Swish** - recognized to be an equal or even better than ReLU, this activation function was created by researchers at Google. Swish is recommended to use in networks with 40+ layers;

- **Gaussian Error Linear Unit (GELU)** - this activation function is well compatible with most NLP models. GELU has a merged functionality of ReLU

(multiplying inputs by 0 or 1, depending if the input value is positive or negative), dropout (sets input value to zero randomly at a rate, therefore prevents overfitting), and zoneout (stochastically multiplies the inputs by 1);

- **Scaled Exponential Linear Unit (SELU)** - this function has an internal normalization property that is adjusting mean and variance and preserves them from previous layers.

3.4 Deep Learning

What is Deep Learning?

Deep learning is a more complex multi-layered structure of algorithms - Neural Networks or also referred to as Deep Neural Networks.

Deep Learning is a key technology nowadays and is attracting lots of attention because of its results. It has improved that much, so it can throw a shade on human performance.

How?

It finally has access to the amount of data needed to be extremely accurate, and do things that were impossible in the nearest past.

Why is it better?

Deep Learning models got their benefits over Machine learning models. In the case of Machine Learning, it is using so-called flat algorithms that could not be applied to raw data, so there is an extra step implemented - Feature Extraction. This step is done by humans and it prepares raw data to be used by Machine Learning algorithms. "Feature Extraction is usually quite complex and requires detailed knowledge of the problem domain. This preprocessing layer must be adapted, tested and refined over several iterations for optimal results." (Artem Opperman, 12 November 2019). In the case of Deep Learning, this step is not needed since it already is a part of the process of Neural Networks and model will recognize unique features by itself. Except that, Deep Learning models keep improving themselves with an increase of the data unlike Machine Learning models, which stop improving after saturation point.

“The analogy to deep learning is that the rocket engine is the deep learning models and the fuel is the huge amounts of data we can feed to these algorithms.” - Andrew Ng, chief scientist in Baidu and one of the leaders in Google Brain project.

3.4.1 Deep Learning in use

Deep Learning apps are used for multiple purposes, such as:

- **Medical Research** - the healthcare industry is using deep learning for accurate and relatively fast results. For example, an advanced microscope was built in order to detect cancer cells, it works with high-dimensional datasets to train the machine and detect them.

- **Electronics** - in electronics deep learning is widely used, smart houses, voice assistants - powered by deep learning.

- **Aerospace and defense** - used in order to research a focus area from satellites, as well as check if the area is fitting for further exploring with troops.

- **Automated driving** - detecting objects such as signs or people walking by became possible with this technology.

- **Industrial automation** - in factories the automation process improves the safety within. It detects when workers are at a close unsafe distance, therefore, performs safety protocols.

3.5 AI vs. Machine Learning vs. Deep Learning vs. Neural Networks

Terms Artificial Intelligence, Deep Learning, Machine Learning, and Neural Networks are usually used interchangeably, but it is not the same thing, they relate but also they have a lot of differences.

The best way to describe them is nesting. The following image describes it visually.

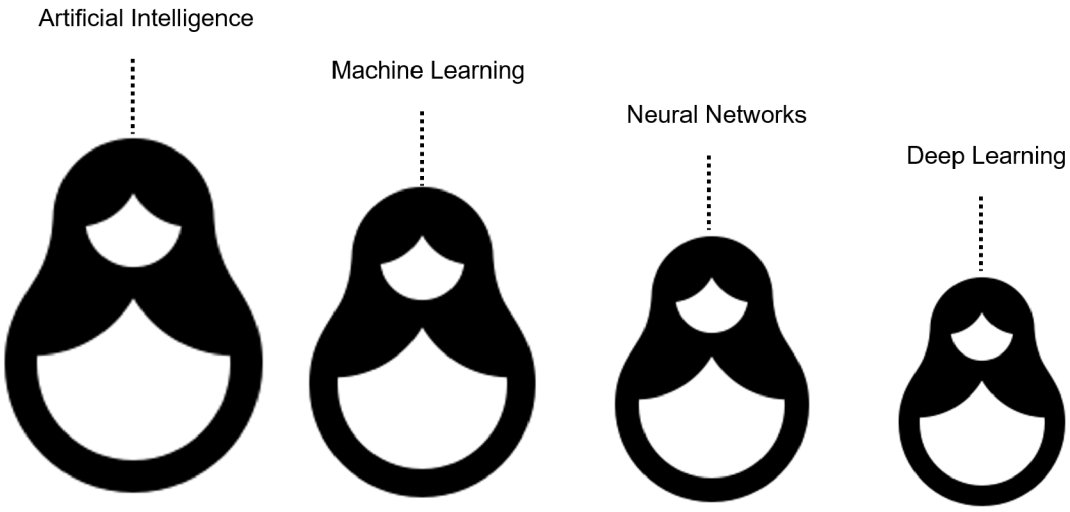

Figure 3. Nesting of AI, ML, NN, DL shown in the image (Eda Kavlakoglu, 2020)

Machine Learning is a subfield of Artificial Intelligence, Neural Networks - of Machine learning, and Deep Learning - of Neural Networks.

When we compare Deep Learning to Neural Networks, it is the number of node layers that makes it "deep", since there must be at least three of them, otherwise, it is a single Neural Network. In case there are more than three depth layers it builds up into a Deep Learning algorithm.

If we speak about the difference between Deep Learning and Machine Learning it is a question of a dataset given, how it learns, and human intervention. Deep Learning works with a vast amount of unstructured data, eliminates the need for humans to check after the outcome, and finds distinguishing features within the dataset by itself, instead, it needs more data to improve accuracy. In the case of Machine Learning, it requires more structured data to learn on, humans to look over, determine hierarchy, explain inputs, set up differences between them so-called “feature extraction”, and so on.

Last but not least is Artificial Intelligence, the most general term that covers them all - means machines that simulate human intelligence.

3.6 ML.NET

ML.NET is an open-source cross-platform machine learning framework for .NET developers. It allows integrating custom-made ML models into .NET apps. Here are some examples of what can be done using ML.NET:

- Sentiment analysis;
- Sales spike detection;
- Product recommendation;
- Customer segmentation;
- Object detection;
- Image classification;
- Fraud detection;
- Sales forecasting;
- Price prediction.

ML.NET does allow to create own ML models without leaving a .NET environment using C# and F#. The platform is beginner-friendly, it provides Auto ML and other tools that allow easy to build and train ML models, and at the same time doesn't require any previous ML experience.

Libraries such as Infer.NET, ONNX, or TensorFlow could be used for additional scenarios making it versatile.

It is said to be more accurate than other ML frameworks, as of the paper - "Machine Learning at Microsoft with ML.NET", 15 May 2019. This paper compares the most popular ML frameworks using a 9GB Amazon dataset.

The performance outcome is following:

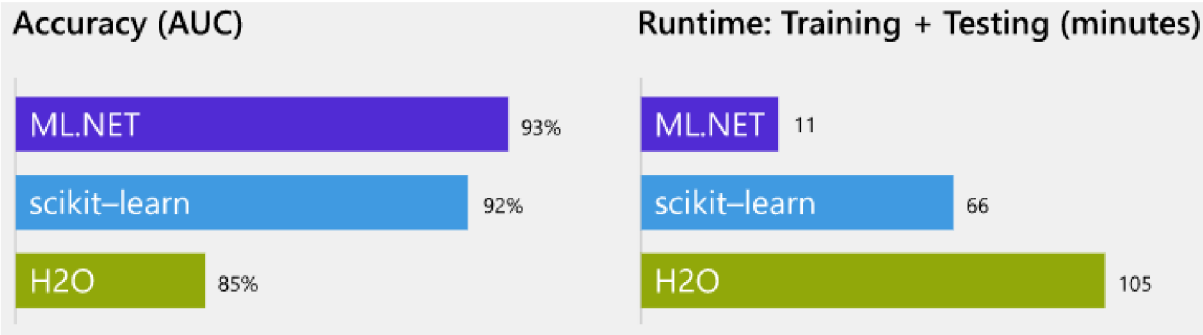

Figure 4. Performance comparison of different ML frameworks by Microsoft (no date)

3.7 .NET platform

.NET is an open-source, cross-platform, free development framework for creating multi-purpose applications. .NET supports languages such as C#, F#, or Visual Basic and offers its own libraries, editors and tools in order to build for web, desktop, mobile, games, and IoT.

Different implementations allow it to run easily on different OS. For example:

- **.NET** - used for websites, servers, and console apps for macOS, Linux, and Windows;
- **.NET Framework** - Windows-focused, supports websites, services, desktop apps, etc.;
- **Xamarin/Mono** - this implementation is used for running apps on all mobile devices.

The platform has extensive libraries such as NuGet, which is built specifically for it with around 90,000 packages.

3.8 TensorFlow

TensorFlow is an end-to-end platform that makes it easy for you to build, deploy ML models, and solve real-life problems.

"TensorFlow has always provided a direct path to production. Whether it's on servers, edge devices, or the web, TensorFlow lets you train and deploy your model easily, no matter what language or platform you use.

Use TensorFlow Extended (TFX) if you need a full production ML pipeline. For running inference on mobile and edge devices, use TensorFlow Lite. Train and deploy models in JavaScript environments using TensorFlow.js. Build and train state-of-the-art models without sacrificing speed or performance. TensorFlow gives you the flexibility and control with features like the Keras Functional API and Model Subclassing API for creation of complex topologies. For easy prototyping and fast debugging, use eager execution.

TensorFlow also supports an ecosystem of powerful add-on libraries and models to experiment with, including Ragged Tensors, TensorFlow Probability, Tensor2Tensor and BERT." - definition given by an official website TensorFlow.com.

3.9 Inception Network

Inception Network is a great achievement in the development on CNN classifiers. Unlike others it doesn't just stack convolutional layers, it adds new Inception modules in order to get faster and better performance as well higher accuracy.

It was first mentioned in a paper “Going deeper with convolutions” (17 September 2014), where it was defined as “...a deep convolutional neural network architecture codenamed Inception, which was responsible for setting the new state of the art for classification and detection in the ImageNet Large-Scale Visual Recognition Challenge 2014 (ILSVRC14). The main hallmark of this architecture is the improved utilization of the computing resources inside the network. This was

achieved by a carefully crafted design that allows for increasing the depth and width of the network while keeping the computational budget constant. To optimize quality, the architectural decisions were based on the Hebbian principle and the intuition of multi-scale processing. One particular incarnation used in our submission for ILSVRC14 is called GoogLeNet, a 22 layers deep network, the quality of which is assessed in the context of classification and detection."

A proof of it's advantage over others is its outstanding breakthrough performance on the ImageNet Visual Recognition Challenge (2014), which is a platform for benchmarking image recognition and detection algorithms.

In order to create a better deep learning model normally you would make it deeper, but this is causing other problems like overfitting(especially with small data) or a need to increase computational power, which is expensive. Paper suggests a solution for this, and solution is to replace fully connected layers to a sparsely connected architecture, meaning increasing not only depth but also width of the network.

Inception v1 or GoogleLeNet was the first implementation of the idea.

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

Figure 5. *GoogLeNet incarnation of the Inception architecture* ("Going deeper with convolutions", 17 September 2014)

Some layers are called "inception" and they are the core idea behind a sparsely connected architecture. "(Inception Layer) is a combination of all those

layers (namely, 1×1 Convolutional layer, 3×3 Convolutional layer, 5×5 Convolutional layer) with their output filter banks concatenated into a single output vector forming the input of the next stage."

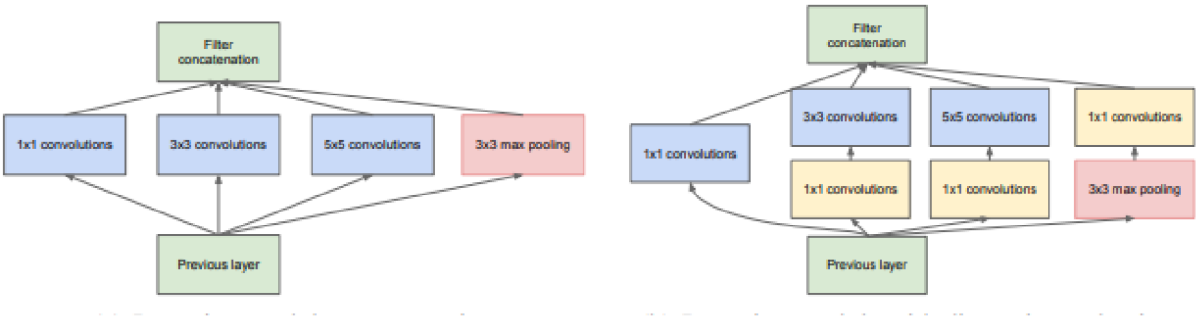

Figure 6. : *Inception module* ("Going deeper with convolutions", 17 September 2014)

Inception layers allow internal layers to pick filter size according to the image therefore get the most relevant information, this explains its effectiveness.

Fun fact, "codenamed Inception, which derives its name from the Network in network paper by Lin et al in conjunction with the famous "we need to go deeper" internet meme". ("Going deeper with convolutions", 17 September 2014)

4. Practical Part

4.1 Goal

The practical part of the thesis will be focused on building and training a model that will be able to put any new unknown item of trash into the following categories: paper, glass, or plastic. The model will be utilised in a simple prototype console application.

4.2 Prerequisites

In order to fulfil this task, following tools and resources will be used:

- Visual Studio 2019 running on Windows 10 as operating system;

- The Inception V1 ML model, which is already pre-trained to classify images into thousand categories and has image processing features needed to build a custom image classifier;

- Image files & CSV file, which pairs labels and images in two columns: “ImageData” and “Label”.

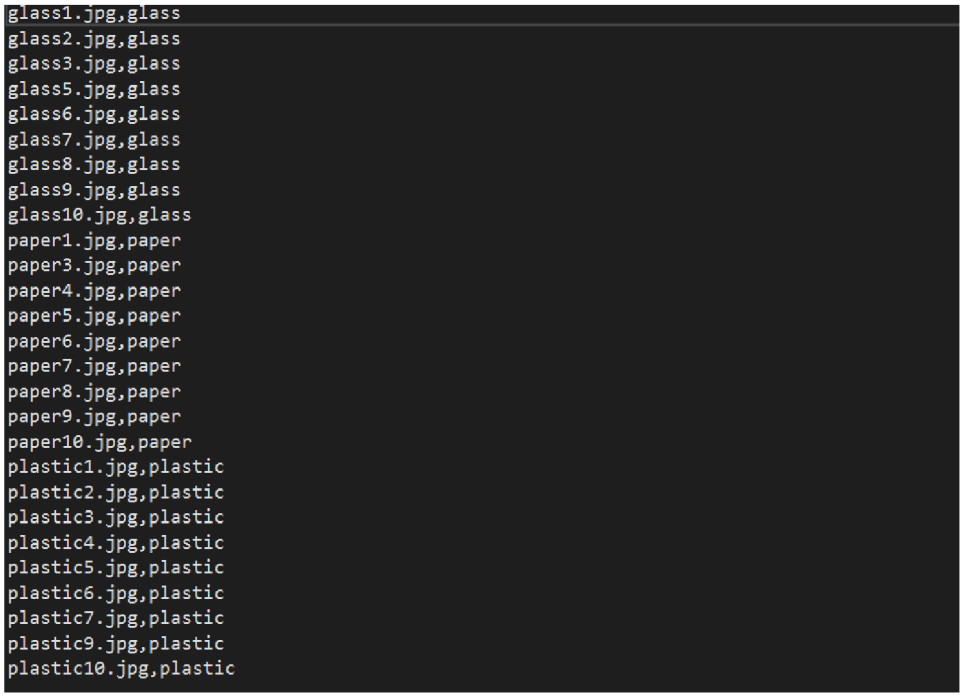

Figure 7. Labeled image files in “labels.csv”

Image files are stored in a folder "images", there are 40 for each category, total 120 images. 21 of them are excluded from the CSV file (7 from each category), for example: "glass4.jpg", "paper2.jpg" and "plastic8" belonging to according categories. These and other files will be used for testing later on.

Figure 8. Example of images used

4.3 Workspace setup

As a first step to create a C# application prototype, .NET Core Console Application is created in Visual Studio 2019. Second, following NuGet packages are installed:

- Microsoft.ML;
- Microsoft.ML.ImageAnalytics;
- SciSharp.TensorFlow.Redist;
- Microsoft.ML.TensorFlow.

Third, moving “images” folder, “labels.csv” file and a “model” folder that contains files needed in order to use the Inception model.Next, in Visual Studio, Solution Explorer for all the added files we change the property of “Copy to Output Directory”, value needs to be “Copy if newer”. That's it, all set up, the environment is ready to be used.

4.4 Creating an application

4.4.1 Libraries

To the top of the program the following “using” statements need to be added in order to use the libraries required.

```
using System;
using Microsoft.ML;
using System.IO;
using System.Linq;
```

Figure 9. “using” statements

4.4.2 Creating additional classes

4.4.2.1 “ImageData” class

The purpose of this class is to load the information from the “labels.csv” file.

For this class an additional “using” statement is added :

- using Microsoft.ML.Data;

Two strings are created:

- “ImagePath” - loads the name of an image file;
- “Label” - loads a label value for images.

```
public class ImageData
{
    [LoadColumn(0)]

    5 references
    public string ImagePath { get; set; }

    [LoadColumn (1)]

    0 references
    public string Label { get; set; }

}
```

Figure 10. “ImageData” class

4.4.2.2 “ImagePrediction” class

“ImagePrediction” class inherits properties of the “ImageData” class. It will be used after the model is trained and ready to use.

For this class, following fields are created:

- “Score” - will predict an accuracy or a confidence percentage;
- “PredictedLabelValue” - contains a value of a label for a predicted image.

Both of them are used to predict and evaluate an input image.

```
public class ImagePrediction: ImageData
{
    2 references
    public float[] Score { get; set; }

    2 references
    public string PredictedLabelValue { get; set; }

}
```

Figure 11. “ImagePrediction” class

4.4.2.3 “InceptionSettings” class

Last one we create an “InceptionSettings” struct, since it only has value type semantics. Here we pass in the constant parameters needed in the Inception model.

```
public struct InceptionSettings
{
    public const int ImageHeight = 224;
    public const int IMageWidth = 224;
    public const float Mean = 117;
    public const float Scale = 1;
    public const bool ChannelsList = true;
}
```

Figure 12. “InceptionSettings” class

4.4.3 Initializing variables In Main

1. First we must initialize a “context” variable with a new MLContext instance, by this a new ML.NET environment is created that is shared across all the objects of the model. Every single ML operation starts with creating an MLContext. It provides everything needed in order to build a pipeline, means it allows the creation of elements for loading data, feature extraction, training, predicting and evaluating.

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

Figure 13. “context” variable

2. Initializing a “data” variable, we specify a file which text data will be loaded from, using the previously created class “ImageData” and “LoadFromtextFile” method. Since data is loaded from a CSV file (Comma Separated Values) - “separatorChar” is set to comma.

```
var data = context.Data.LoadFromTextFile<ImageData>("./labels.csv", separatorChar: ',');
```

Figure 14. “data” variable

4.4.3.1 Pipeline.

Pipeline is a set of chained methods, no execution is happening at this step, until the Fit method is called. Data cannot be used raw , therefore it must be transformed, by using the “Transforms” property. Data transformation helps to convert data from a raw state into a ready for analysis state, making sure it is of maximum quality.

First transformation happens to labels, since value is categorical, so it has to be transformed into numerical form. The images are loaded then, as input from the “images” folder, using “ImagePath” from the “ImageData” class. Next transformation is resizing them using “ImageHeight” and “ImageWidth” from the “InceptionSettings”

class, then stored as new input. After pixels are extracted, they are stored in the channel list, in a form of all values for one color for all pixels, then the same for another color and so on.

After all the transformations TensorFlow is loaded from the "model" folder, SoftMax activation function is used, "addBatchDimensionInput" is set to true, meaning, if data has unknown shape but model requires data to have batch dimension too.

Next is a trainer of the classification model with a Limited-memory Broyden–Fletcher–Goldfarb–Shanno algorithm (L-BFGS), counting new label values. This is a popular optimization algorithm for parameter estimation that provides faster convergence towards the minimum.

At last numerical "PredictedLabelValue" is counted and converted back to categorical as "PredictedLabel".

```
var pipeline = context.Transforms.Conversion.MapValueToKey("LabelKey", "Label")
    .Append(context.Transforms.LoadImages("input", "images", nameof(ImageData.ImagePath)))

    .Append(context.Transforms.ResizeImages("input", InceptionSettings.IMageWidth,
    InceptionSettings.ImageHeight, "input"))

    .Append(context.Transforms.ExtractPixels("input", interleavePixelColors:
    InceptionSettings.ChannelsList, offsetImage: InceptionSettings.Mean))

    .Append(context.Model.LoadTensorFlowModel("./model/tensorflow_inception_graph.pb")
        .ScoreTensorFlowModel(new[] { "softmax2_pre_activation" }, new[] { "input" },
        addBatchDimensionInput: true))

    .Append(context.MulticlassClassification.Trainers.LbfgsMaximumEntropy("LabelKey",
    "softmax2_pre_activation"))

    .Append(context.Transforms.Conversion.MapKeyToValue("PredictedLabelValue", "PredictedLabel"));
```

Figure 15. "pipeline" variable

3. Model is created by fitting the data into the pipeline.

```
var model = pipeline.Fit(data);
```

Figure 16. "model" variable

4. Now the data is being transformed.

```
Console.WriteLine("\n------------Evaluate-----------------");

var evaluatePredictions = model.Transform(data);
```

Figure 17. “evaluatePredictions” variable

5. By initializing the “metrics” variable, the performance of the model will be evaluated using the “.Evaluate()” method. It will return the performance metrics.

   Log Loss - is a metric that has a value that defines an accuracy of the prediction. The lesser the number(should be as close to zero as it can be) the more accurate the classifier is. Same thing will be done to each class accordingly.

```
var metrics = context.MulticlassClassification.Evaluate(evaluatePredictions, labelColumnName: "LabelKey",
    predictedLabelColumnName: "PredictedLabel");

Console.WriteLine($"Log Loss - {metrics.LogLoss}");
Console.WriteLine($"Per class Log Loss - {String.Join(',', metrics.PerClassLogLoss.Select(l => l.ToString()))}");
```

Figure 18. “metrics” variable

6. Next the “.CreatePredicitionEngine” method was used to create a prediction engine. Input data is defined by “ImageData” class, and the output by “ImagePredicition” class.

```
var predictionFunction = context.Model.CreatePredictionEngine<ImageData, ImagePrediction>(model);
```

Figure 19. “predictionFunction” variable

7. The performance of the model is assessed by trying to make a single prediction to an image that wasn’t previously manually labeled - “glass4.jpg”. Later on in Results will be done two more single predictions for items “plastic8.jpg” and “paper2.jpg” in order to try and check it a few more times.

```
var singlePrediction = predictionFunction.Predict(new ImageData
{
    ImagePath = Path.Combine(Environment.CurrentDirectory, "images", "glass4.jpg")
});
```

Figure 20. “singlePrediction” variable

8. In this step the outcome will be displayed. The model will predict which category the “glass4.jpg” belongs to and at what score. The closer score to 1 - the better.

```
Console.WriteLine("\n------------Predictions for unknown items----------------");

Console.WriteLine($"Item {Path.GetFileName(singlePrediction.ImagePath)} belongs to a waste category - " +
    $" {singlePrediction.PredictedLabelValue} " +
    $"with a score of {singlePrediction.Score.Max()}");

Console.ReadLine();
```

Figure 21. Displaying an outcome

4.5 Performance assessment

Application is complete, now it is time to run it in order to check its functionalities and predictions. There were no visible problems during the execution, fast loading and acceptable results. Let's take a closer look at the outcomes.

4.5.1 Outcomes

4.5.1.1 Log Loss

Results of the total Log Loss and per class Log Loss are in the same range without any drastic difference. All of them are close to zero, meaning the model is acceptably accurate, though the dataset is relatively small.

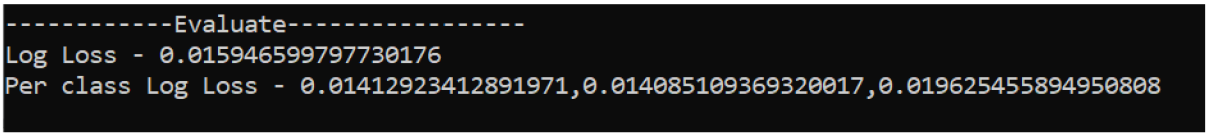

Figure 22. Results of the Log Loss and Log Loss per class

4.5.1.2 Testing predictions for the unknown items

15 items out of 21 were recognized correctly with a score 0,92+ which is a confident prediction.

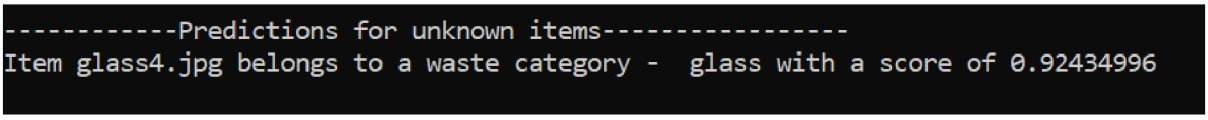

Figure 23. Prediction for an unknown item “glass4.jpg”

3 items out of 21 were recognized correctly with a score ranging from 0,71 to 0,88 making it an average confidence.

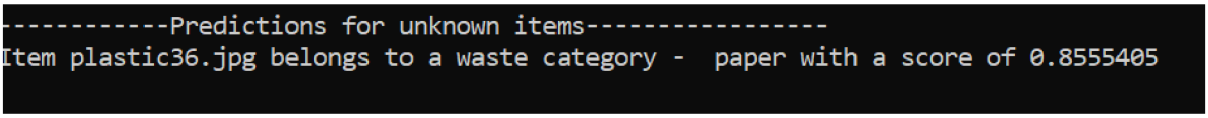

Figure 24. Prediction for an unknown item “ “plastic36.jpg”

3 predictions out of 21 were wrong.

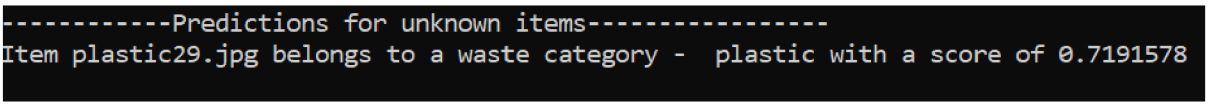

Figure 25. Prediction for an unknown item “plastic29.jpg”

Overall 86% samples from the training dataset were correctly predicted with a high recognition score.

5. Results

The result of this bachelor thesis is a fully functional program that has a relatively good success rate considering a small sample size. Model can definitely be improved by adding more data to it as well as other custom features or some parts of it can be used for other purposes.

This thesis shows the use case of implementation of a pre-trained TensorFlow model, which can be helpful in such a sensitive topic like global pollution, where it can be used to improve the situation overall by automating the waste separation process.

The work was done using a C# programming language, ML.NET libraries as well as a pre-trained TensorFlow model, source code is attached to the thesis. The main “Program.cs” file is used to run it.

During the building process no major problems have occurred, the model works as intended with relatively good outcomes. For a practical application more data is needed in order to improve the model and to cover the variability in the image set. Although this model is not ready for a real-world application, it is indicative of possibilities CNN’s can offer.

6. Conclusion

This thesis focused on a real implementation of an existing Machine Learning model with a full theoretical background for the technologies that are used or connected to the topic. The Literature Review part explains in great detail the history, purposes and functionalities, underlying and related technologies of fields such as Artificial Intelligence, Machine Learning, and Deep Learning. It also compares them in order to have a better understanding of their role.

Object Recognition and Classification is described and performed in the practical part of this thesis on a case study of waste recognition application prototype with a step-by-step guidance through all the stages of implementation.All the

features of the model and application are described and their purposes are explained. The model has been tested and is working as intended.

We live in an era of Technologies and these technologies can be used for multiple purposes as described in this work. It allows us to improve the quality of our life relatively easily. Automation saves us not only from physical exhaustion but also harm, and enables us towards pursuing new fields and picking knowledge and self-improvement over hard physical work.

7. References

Figure 1. Adnan Sheikh (2021) [Machine Learning Process]. Available at: Learn Cloud Bits (Accessed: 29 November 2021)

Figure 2. Akizur Rahman (November 2019) *The basic ANN architecture.* Available at:The basic ANN architecture. Neuron is the smallest processing component... | Download Scientific Diagram (researchgate.net) (Accessed: 29 November 2021)

Figure 3. Eda Kavlakoglu (2020) [Nesting of AI, ML, NN, DL]. Available at: AI vs. Machine Learning vs. Deep Learning vs. Neural Networks: What's the Difference? | IBM(Accessed: 29 November 2021)

Figure 4. Microsoft (no date) [Performance comparison of different ML frameworks]. Available at: ML.NET | Machine Learning made for .NET (microsoft.com) (Accessed: 12 November 2021)

Figure 5. Christian Szegedy, Wei Liu, Chapel Hill Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich (17 September 2014) *Going deeper with convolutions.* Available at: 1409.4842v1.pdf (arxiv.org) (Accessed: 29 November 2021)

Figure 6. Christian Szegedy, Wei Liu, Chapel Hill Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich (17 September 2014) *Going deeper with convolutions.* Available at: 1409.4842v1.pdf (arxiv.org) (Accessed: 29 November 2021)

IBM Cloud Education (3 June 2020) *What is Artificial Intelligence (AI)?* Available at: What is Artificial Intelligence (AI)? | IBM (Accessed: 12 September 2021)

IBM Cloud Education (27 May 2020) *AI vs. Machine Learning vs. Deep Learning vs.Neural Networks: What is the difference?* Available at: AI vs. Machine Learning

vs. Deep Learning vs. Neural Networks: What's the Difference? | IBM (Accessed: 14 September 2021)

Artem Oppermann (12 November 2019) *What is Deep Learning and how does it work?* Available at: What is Deep Learning and How does it work? | Towards Data Science (Accessed: 18 September 2021)

John Burke (March 2021) *What is a Neural Network? Explanation with examples.* Available at: What is an Artificial Neural Network (ANN)? (techtarget.com) (Accessed: 02 October 2021)

Jo Best (9 September 2013) *IBM Watson: the inside story of how the Jeopardy-winning supercomputer was born, and what it wants to do next*. Available at: IBM Watson: The inside story of how the Jeopardy-winning supercomputer was born, and what it wants to do next - TechRepublic (Accessed: 08 October 2021)

Zeeshan Ahmed, Saeed Amizadeh, Mikhail Bilenko, Rogan Carr, Wei-Sheng Chin, Yael Dekel, Xavier Dupre, Vadim Eksarevskiy, Eric Erhardt, Costin Eseanu, Senja Filipi, Tom Finley, Abhishek Goswami, Monte Hoover, Scott Inglis, Matteo Interlandi, Shon Katzenberger, Najeeb Kazmi, Gleb Krivosheev, Pete Luferenko, Ivan Matantsev, Sergiy Matusevych, Shahab Moradi, Gani Nazirov, Justin Ormont, Gal Oshri, Artidoro Pagnoni , Jignesh Parmar, Prabhat Roy, Sarthak Shah, Mohammad Zeeshan Siddiqui, Markus Weimer, Shauheen Zahirazami, Yiwen Zhu (15 May 2019) *Machine Learning at Microsoft with ML.NET*. Available at: Machine Learning at Microsoft with ML.NET (arxiv.org) (Accessed: 23 November 2021)

TensorFlow (no date) *Why TensorFlow?* Available at: Why TensorFlow (Accessed: 20 November 2021)

Tutorial, Jon Wood (2 September 2019) *Image Classification with Transfer Learning in ML.NET using a TensorFlow Model*. Available at: Image Classification with Transfer Learning in ML.NET using a Tensorflow Model - YouTube (Accessed: 15 November 2021)

Microsoft (29 September 2021) *Tutorial: Train an ML.NET classification model to categorize images*. Available at: Tutorial: ML.NET classification model to categorize images - ML.NET | Microsoft Docs (Accessed: 16 November 2021)

8. Appendix

Files attached to the thesis:

- images - folder with images used;
- model - contains Inception model;
- labels.csv - contains labeled files;
- test-labels.csv - contains test labels for training;
- program.cs - main script of the program;
- bin, obj, transferLearning, ImageData.cs, ImagePrediction.cs, InceptionSettings.cs, transferLearning.csproj, transferLearning.sln - files created by Visual Studio.