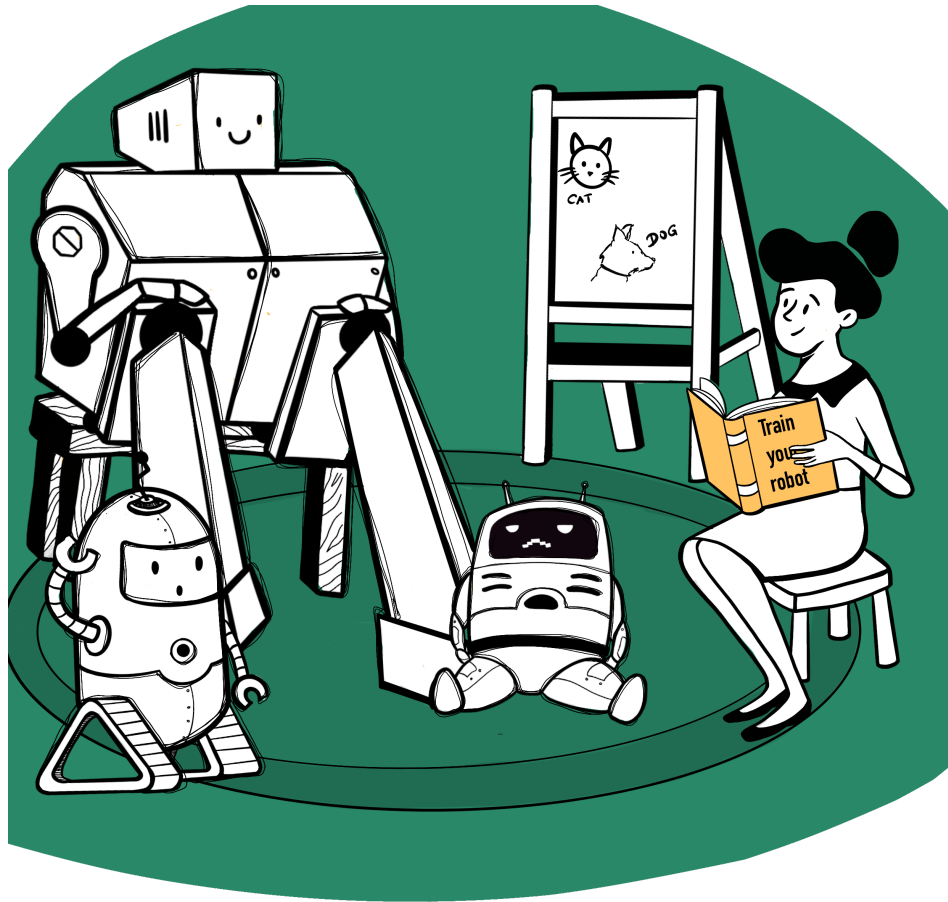


Module 4

Supervised Learning

"The **model** consists of **parameters** that are not chosen by the programmer but **adapted by the algorithm** in a process called **training**."



About the Module

This module is about **Supervised Learning (SL)**. The goal is to provide the students with a basic understanding of what **SL** is, what it can and cannot do as well as how one can train **SL**-algorithms. It focuses more on the practical side, therefore students will train their own algorithms and experience the possibilities and problems first-hand. The underlying technical part will be skipped for the most part, as there are other modules for understanding specific methods like **Neural Networks**, and this knowledge is not required for a basic understanding of how **SL** works or how it can be used.

Objectives

Students will be able to...

- ...explain **SL** as a mapping function
- ...name the value and requirements of training data
- ...estimate if an application is using **SL**
- ...name problems and limitations of **SL**
- ...train their own **SL** model and use it inside an application

Agenda

Time	Content
30 min	Introduction
30 min	Exercise - Train your first model
15 min	Theory - Reliance on data
30-60 min	Exercise - Game controller
30 min	Discussion + Quiz - Possibilities and limitations

Introduction

The Introduction is all about students becoming familiar with basic concepts and terms used in **Supervised Learning (SL)**. It is based on the slides, which introduce not only common applications but the general idea of what a **SL** algorithm does and how a **SL** algorithm can be trained.

Real World Examples

(Slides 2 - 8)

In the first slides several applications for **SL** algorithms are introduced, which should make it clear that it is used in a lot of different applications.

Prime examples for **SL** are **classifying** images and **detecting** objects in images. Face detection for instance, can be used to auto-focus a camera or create automatic labels with names in case of social media sites. The goal of object recognition is to detect and identify different objects inside an image. Therefore, usually the first step is segmentation, where the image is clustered into different areas, which then are classified with one of the available labels. All these steps nowadays are mostly done using **SL** and **Deep Neural Networks (DNN)**. Even in production industries **SL** and image classification are used, for instance to detect defects or assembly mistakes automatically. This is one of the areas where **AI** is replacing more and more human workers due to lower costs and sometimes even due to making less errors. In areas like medicine however, algorithms for detecting illnesses are used in conjunction with doctors, to help them to come to the correct conclusion. This is done for instance by pointing out potential cancer spots on scans, which might have been overlooked otherwise.

The next example is speech recognition and **Natural Language Processing (NLP)**. Instead of an image, sound waves are used to distinguish between spoken words and other sounds. Once words are recognized, they can be transcribed, which is used for instance for automatic subtitle generation. Furthermore, sentences can be analysed for their meaning, so that applications like digital assistants (Alexa, Siri, Google Now, ...) can understand complex commands and queries and answer/act in a meaningful way. This way of analyzing sentences/texts is also used in spam filters to distinguish between normal mail and spam mail.

Another usage of **SL** algorithms is predicting values like the cost of a house or the revenue of a planned movie, based on various information like size of the house or actors in the movie. In this case a **SL** algorithm is trained using data on similar houses and their costs and then used to predict the value of new homes.

Finally, **SL** algorithms are also used to classify users into different profiles, to provide customized advertisements. As the advertisement corresponds more to the interests of the user, there is a higher chance he or she will click on the link which results in a higher chance of selling the product.

What Supervised Learning Algorithms really do

(Slides 9 - 13)

After all the examples, Supervised Learning algorithms are introduced as mapping functions. The main point here is that they do a very simple thing, given an input they produce an output. There is no magic **AI** decision making going on, it is just a mathematical formula that produces a clear result. It is a good idea to look at the examples from before and talk about what could be the input, and what would be the output. For example, when the input is an image, the output could be positions and labels of different objects, or just a label like 'defect' or 'ok'. When the input is a sound sample, the output could be the corresponding words and when the input is general information about a movie production, the output could be the expected revenue. These types of problems are called **classification** (labelling data) and **regression** (predicting values).

It is important to mention that the output is usually not binary, but gradually. For instance, given an image of a cat, the algorithm would be 99.5% certain that it is a cat, 0.4% certain that it is a dog and 0.1% certain that it is a fish.

How to train Supervised Learning Algorithms

(Slides 14 - 26)

In the last section the process of **training** is introduced. It is based on the idea that the developer provides the algorithm with examples (e.g., this is a dog) and the Supervised Learning program adapts its internal **model** (e.g., variables in a formula) to produce the correct output. Additionally, the concept of **trainings-** and **test-data** is introduced.

There are five important definitions here:

Supervised Learning Algorithm

An **algorithm** is a step-by-step description of a task. In this case it defines exactly how the **SL** process works, how it learns as well as what **parameters** are used and how they influence the result. It is often compared to a cooking recipe, where there are detailed instructions which, when followed correctly, lead to a tasty result.

Parameters

Parameters are data that can be used to modify the result or working process of an algorithm. To continue with the cooking example, a parameter of a cake could be how sweet it should be. The cook then decides for the amount of sugar and follows the recipe accordingly. In **SL** parameters are used in the training process, where the programmer has to decide in which way (how many iterations, internal structure, what to focus on, ...) the algorithm has to learn.

Supervised Learning Model

The model consists of parameters that are not chosen by the programmer but **adapted by the algorithm** in a process called **training**. In the cooking example, these could be a list of ingredients with their corresponding amount, which the algorithm then has to optimize during training to get the desired (e.g., tasty) result. Especially in image classification **Neural Networks** are often used as the underlying model, therefore models are often just called **networks**.

Labelled Data

A set of data that contains not only the input but also the output of the **SL** algorithm. For example, this can be a few hundred images of cats and dogs with their corresponding labels 'cat' and 'dog'.

Trainings- and Test-Data

Usually, one starts by collecting a lot of **labelled data**, which then is split into two sets, the trainings-data and the test-data. The trainings-data is used to train the model while the test-data is used to test its accuracy. It is important that this is **different** data, to ensure that the model is general enough and not fixated on the specifics of the trainings-data.

Once these terms are clear, the process of training a model is quite simple:

1. Collect a set of **labelled data** and split it into two sets (training and test)

2. **Initialize your model** with any values (often random) and select a set of **training parameters**
3. **Repeat** with the training-data (as often as defined by the parameters)
 - Given an input, **adapt** the values inside the model to more closely match the given output
4. **Test your model** by using test-data to check its accuracy
5. **Repeat from the beginning**, modifying the labelled data or training parameters, until you are satisfied with your model

Questions and What's next

(Slide 26)

At this point the students should know the basics of how **SL** algorithms work. In the upcoming exercises they will then try this process themselves and experience the **problems** that can occur and how sensitive these algorithms are to their trainings-data.

While this course does not go into the technical details, it is worth mentioning that depending on the problem, there are multiple algorithms that can be used for **SL**. Most people have heard about (deep) neural networks, which are used for many problems, especially when the data is quite complex, like in image or voice recognition. But there are many more algorithms like decision-trees, regression curves, nearest neighbor and more.¹

Material

-  SL - Introduction.pdf

References

1. <https://builtin.com/data-science/supervised-machine-learning-classification>

Train your first model

The goal of this exercise is to give the students a first impression of how the **training** of a **SL** algorithm works and what **difficulties** can occur. The main takeaway should be that training is heavily dependent on the **trainings-data** and that it is generally a process that needs many iterations to lead to a useful result.

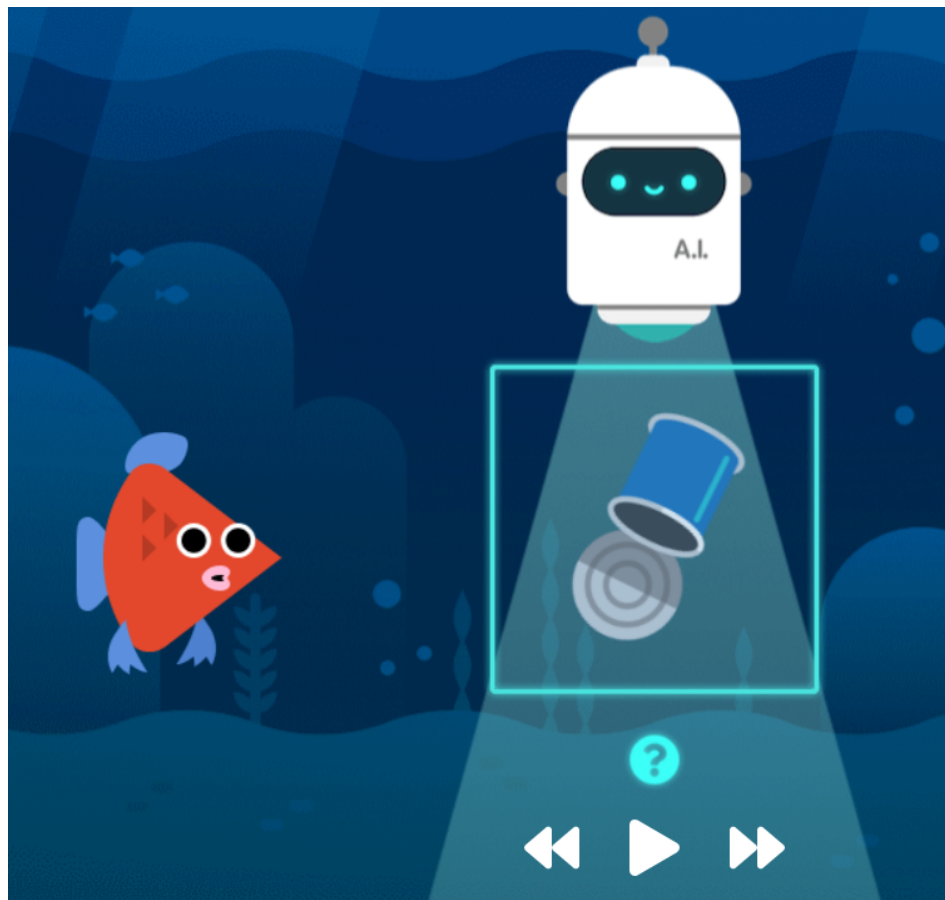
There are two possible exercises:

AI for Oceans which is an online tool and the **Classification Game** which is a hands-on activity.

AI for Oceans

This exercise uses the ocean scanning exercise from the course **AI for Oceans**. Using a **web browser**, students have to train a robot to differentiate between fish and garbage as well as between different kinds of aquatic animals.

Exercises / AI for Oceans



Classification Game

This exercise puts the students in the shoes of an **SL** algorithm so that they can experience the difficulties of classifying images of cats and dogs in this **pen-and-paper** exercise.

Exercises / Classification Game



Objectives

Students will be able to...

...explain the process of training a **SL** algorithm

...explain how the algorithms work by focusing on various features

...explain how the system is very dependent on its trainings-data

Reliance on data

This chapter is about **trainings-data** and the impact and importance of the **quality** of this data. While the examples in this chapter will focus on image recognition, all the discussed problems still apply to all forms of Supervised Learning. The chapter is based on **these slides**.

Overfitting and Underfitting

(Slides 2-14)

The two biggest problems when training a **SL** algorithm are **overfitting** and **underfitting**.

Overfitting is the problem that a model works well **during training** but provides poor results on **different data**. It occurs when the training set is either too specific or provides prominent misleading features.

Underfitting on the other hand means that the model is not learning anything useful at all. It occurs when the input data is **insufficient** in number or **too general** and no related features can be found. Let's look at some examples.

Imagine you want to create a model that differentiates between different people by looking at portrait pictures. You ask your friends to give you a few hundreds of portrait images, recorded with their webcam and use them to train your algorithm. While the test results are very promising, you realise that when you test your model by inviting your friends to your home, it barely detects anyone correctly.

This is a classic example of **overfitting**, where under specific circumstances the model works fine, but it fails horribly once one changes the surrounding environment. A reason for failing could be that the model did not learn the features of faces as expected, but focused on plants or shelves in the background. These background features are often quite distinct and can even lead to the algorithm ignoring the face and only recognizing the background, therefore the input data was too specific.

An easy way to reduce the problem is to provide a **more diverse dataset** (different scenery, lighting, camera angles, head positions, zoom levels, haircuts, more images, ...), so that specific unwanted features occur only rarely. This

problem can be further reduced by **processing the data** to reduce unnecessary features. Transforming everything to grayscale images can reduce the reliance on colours, as can normalizing the saturation of all images. Geometric transformations like scaling, rotating or flipping images can also provide a much bigger dataset that is more general. It is important though, to **keep your data in mind**, if you want to detect different coloured pens it might be a bad idea to switch to grayscale images, and if you want to detect certain gestures, flipping an image can be detrimental.

For **underfitting** let's look at the example of predicting movie revenue.

Imagine you have gathered data on thousands of movies about the genre, actors and revenue. You then train your model based on this data but realize, no matter what test data you select, the prediction is quite off most of the time.

This means your model has basically learned nothing. This is an example of **underfitting**, as either your training data did not have enough (or relevant) information (features) to predict anything useful or your training parameters are way off.

In this example the inclusion of **more features** like the release year, the budget and budget allocation or the amount of advertisement could overcome this problem and lead to a more accurate prediction. **Underfitting** also occurs when you just don't have enough data and therefore too little information that can be used for prediction or classification.

Underfitting and overfitting are both prominent and usually easy to detect (but not necessarily easy to fix...). Next, we will talk about the more subtle problem of **biases**.

Biases and Fairness

(Slides 15-19)

While **underfitting** and **overfitting** are technical and clearly measurable problems, **biases** are much harder to detect. Let's start with a real example.

*In 2014 Amazon, like many other big companies, tried to automate their recruitment process by creating an AI to review job applications and find the most talented / suitable workers. After one year of development however, they realized a big flaw: their algorithm **discriminated against women!** It turned out*

that there was no evil intention, but the trainings-data of previously hired workers was hugely biased for men, as most employees were male at this time. While this problem was fixed by carefully selecting the applications used for training, it could not be guaranteed that the algorithm didn't have any other, currently not visible biases, so the project was disbanded in 2018.¹

*Another example is Microsoft's chatbot Tay2, which adapted **inflammatory and racist speech** from users on Twitter and had to be taken down after only 16 hours.²*

In principle it is possible to create models that are fair, but depending on the subject it can be a **very hard problem**. The problem ranges from service bots not recognizing dialects of some minorities of people to classification algorithms not detecting people of colour as humans due to limited or exclusive training data. These problems are not only technical but ethical as they raise the question what decisions we should let machines decide. The **Ethics module** investigates this question in more detail. Another good resource is **this video**.

Biases and fairness are currently widely researched fields and no easy solution can be given. All you can do is to be careful when creating training-data for your model to train on. This is also reflected in a common saying in computer science: "the algorithm is only as good as it's data" or in simpler words: "garbage in, garbage out".

Training Time and Transfer Learning

(Slides 20–23)

As demonstrated in the topics above, the quality of the model is highly dependent on its chosen data and therefore also of the **size of its data**. To create an unbiased image classification model that accurately detects objects in various situations thousands or even millions of images have to be used, which also increases the time needed for training the model from hours to weeks.

To counteract this problem **transfer learning**³ can be used to benefit from the experience of already well trained (**pre-trained**) models. These models usually focus on more generic tasks, like finding features in images, which can then be used either as a **starting point** for training, or as a **pre-process step** to reduce the complexity of the input data. In the first case, a network that is already used in image classification can sometimes be used as a base to train a network to

classify a different set of images (like from detecting forest fires in satellite images to detecting fractures in X-ray images). In the second case, a network that can detect features like edges or geometric patterns can be used as a first step and the new network can be trained on these features instead of the raw image data. In both cases not only the training time can be drastically reduced, but also the quality of the result increased, especially when working with very small sample sizes like 50 pictures.

The next exercise uses such pre-trained networks to be able to train image classification networks inside the browser in a matter of seconds to minutes while only requiring a few hundred images for good results.

Material

-  SL - Reliance on data.pdf

References

1. <https://www.theguardian.com/technology/2018/oct/10/amazon-hiring-ai-gender-bias-recruiting-engine>
2. <https://www.theguardian.com/technology/2016/mar/24/tay-microsofts-ai-chatbot-gets-a-crash-course-in-racism-from-twitter>
3. <https://machinelearningmastery.com/transfer-learning-for-deep-learning/>

Game controller

In this exercise, students will train a real image classification model to detect different directional commands. This model will then be used to control and play the game Snake.

The goal of this exercise is to let the students train their own real time image classification model. During this process, many of the previously

discussed problems will occur and students have to try to make their model as robust as possible. To demonstrate further the difficulties in real applications the used camera should be changing direction/position to highlight problems in their training sets (like background objects). In the end the students will have a working model that provides directions and can be used to control the game Snake (or, for very advanced students, in any other application they use TensorFlow in).

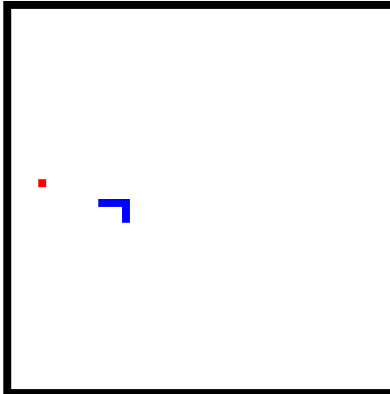
The time required for this exercise can vary wildly, from 30 minutes for a short test to up to multiple hours, when students want to create a good working (general) model.

Exercises / Game Controller

Objectives

Students will be able to...

- ...train a real **SL** image classification model
- ...apply their knowledge about how to get the correct data
- ...realize that teaching the model doing 'the right thing' is one of the main difficulties in **SL**



up: 0.01
down: 0.00
left: 0.72
right: 0.25
none: 0.01
direction: left

Load an **image-model** created with

Your model has to include the class

[load image-model from zip-file](#)

Possibilities and limitations

Now that the students have a good understanding of how Supervised Learning works and what its problems are, it is time to summarize everything and look at real world applications. While the line between different machine learning methods can be quite fuzzy nowadays (as for example classical **SL** can be improved using reinforcement learning), this chapter is about detecting the use of **SL** in different applications and its limitations.


This chapter takes the form of a quiz using (mostly) **true or false** questions on **slides**. Students can participate live using for hand signs or coloured/numbered cards. The quiz should not only be about getting the right answers but **understanding** the problems and also **arguing** why other answers might be correct too. Therefore, it is encouraged to not just show the answers but to talk about them in a short discussion. Some questions contain topics not previously discussed, to provide the possibility for students to show how well they can transfer their knowledge to unknown domains.

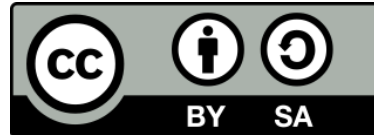
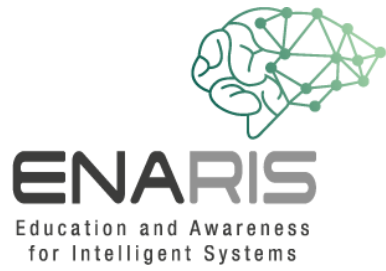
Objectives

Students will be able to...

- ...explain what **SL** learning can and cannot do
- ...recognize which applications can make use of **SL**

Material

-  SL - Quiz.pdf



EUROPEAN UNION

